FUNDAMENTALS OF IoT

1. INTRODUCTION TO IoT

- Today the Internet has become ubiquitous, has touched almost every corner of the globe, and is affecting human life in unimaginable ways.
- We are now entering an era of even more pervasive connectivity where a very wide variety of appliances will be connected to the web.
- One year after the past edition of the Clusterbook 2012 it can be clearly statedthat the Internet of Things (IoT) has reached many different players and gainedfurther recognition. Out of the potential Internet of Things application areas,Smart Cities (and regions), Smart Car and mobility, Smart Home and assistedliving, Smart Industries, Public safety, Energy & environmental protection,Agriculture and Tourism as part of a future IoT Ecosystem (Figure 1.1) haveacquired high attention.

IoT Ecosystem.

- We are entering an era of the “Internet of Things” (abbreviated as IoT).There are 2 definitions: First one is defined by Vermesan and second by Pe˜na-L´opez

  1. The Internet of Things as simply an interaction between the physical and digital worlds. The digital world interacts with the physical world using a plethora of sensors and actuators.

  2. Another is the Internet of Things is defined as a paradigm in which computing and networking capabilities are embedded in any kind of conceivable object.

- We use these capabilities to query the state of the object and to change its state if possible.
- In common parlance, the Internet of Things refers to a new kind of world where almost all the devices and appliances that we use are connected to a network.
- We can use them collaboratively to achieve complex tasks that require a high degree of intelligence.
- For this intelligence and interconnection, IoT devices areequipped with embedded sensors, actuators, processors, andtransceivers.
- IoT is not a single technology; rather it is anagglomeration of various technologies that work together intandem.
- Sensors and actuators are devices, which help in interactingwith the physical environment.
- The data collected by thesensors has to be stored and processed intelligently in order toderive useful inferences from it.
- Note that we broadly definethe term *sensor*; a mobile phone or even a microwave ovencan count as a sensor as long as it provides inputs about itscurrent state (internal state + environment).
- An *actuator* is adevice that is used to effect a change in the environment suchas the temperature controller of an air conditioner.
- The storage and processing of data can be done on theedge of the network itself or in a remote server.
- If any preprocessingof data is possible, then it is typically done at eitherthe sensor or some other proximate device.
- The processeddata is then typically sent to a remote server.
- The storageand processing capabilities of an IoT object are also restrictedby the resources available, which are often very constraineddue to limitations of size, energy, power, and computationalcapability.
- As a result the main research challenge is toensure that we get the right kind of data at the desired levelof accuracy.
- Along with the challenges of data collection, and handling, there are challenges in communication aswell.
- The communication between IoT devices is mainlywireless because they are generally installed at geographicallydispersed locations.
- The wireless channels often have high rates of distortion and are unreliable.
- In this scenario reliablycommunicating data without too many retransmissions is animportant problem and thus communication technologiesare integral to the study of IoT devices.
- We can directly modify the physical world through actuators or we may do something virtually. For example,we can send some information to other smart things.

- The process of effecting a change in the physical world is often dependent on its state at that point of time. This is called *context awareness*. Each action is taken keeping in consideration the context because an application can behave differently in different contexts.
- For example, a person may not like messages from his office to interrupt him when he is on vacation. Sensors, actuators, compute servers, and the communication network form the core infrastructure of an IoT framework. However, there are many software aspects that need to be considered.
- First, we need a middleware that can be used to connect and manage all of these heterogeneous components. We need a lot of standardization to connect many different devices.
- The Internet of Things finds various applications in health care, fitness, education, entertainment, social life, energy conservation, environment monitoring, home automation, and transport systems.

1.2 TECHNOLOGIES INVOLVED IN IOT DEVELOPMENT: INTERNET/WEB AND NETWORKING BASICS OSI MODEL

- Networking technologies enable IoT devices to communicate with other devices, applications, and services running in the cloud.
- The internet relies on standardized protocols to ensure communication between heterogeneous devices is secure and reliable.
- Standard protocols specify rules and formats that devices use to establish and manage networks and transmit data across those networks.
- Networks are built as a “stack” of technologies. A technology such as Bluetooth LE is at the bottom of the stack.
- While others such as such as IPv6 technologies (which is responsible for the logical device addressing and routing of network traffic) are further up the stack. Technologies at the top of the stack are used by the applications that are running on top of those layers, such as message queuing technologies.
- This article describes widely adopted technologies and standards for IoT networking. It also provides guidance for choosing one network protocol over another. It then discusses key considerations and challenges related to networking within IoT: range, bandwidth, power usage, intermittent connectivity, interoperability, and security.

NETWORKING STANDARDS AND TECHNOLOGIES

- The Open Systems Interconnection (OSI) model is an ISO-standard abstract model is a stack of seven protocol layers.
- From the top down, they are: application, presentation, session, transport, network, data link and physical. TCP/IP, or the Internet Protocol suite, underpins the internet, and it provides a simplified concrete implementation of these layers in the OSI model.

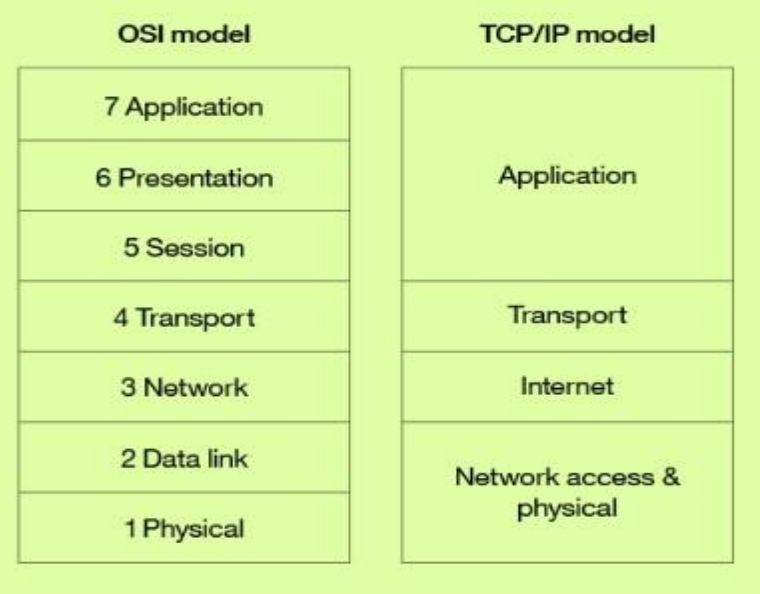


Figure 1. OSI and TCP/IP networking models

The TCP/IP model includes only four layers, merging some of the OSI model layers:

- **Network Access & Physical Layer**

  This TCP/IP Layer subsumes both OSI layers 1 and 2. The physical (PHY) layer (Layer 1 of OSI) governs how each device is physically connected to the network with hardware, for example with an optic cable, wires, or radio in the case of wireless network like wifi IEEE 802.11 a/b/g/n). At the link layer (Layer 2 of OSI), devices are identified by a MAC address, and protocols at this level are concerned with physical addressing, such as how switches deliver frames to devices on the network.

- **Internet Layer**

  This layer maps to the OSI Layer 3 (network layer). OSI Layer 3 relates to logical addressing. Protocols at this layer define how routers deliver packets of data

between source and destination hosts identified by IP addresses. IPv6 is commonly adopted for IoT device addressing.

- **Transport Layer**

  The transport layer (Layer 4 in OSI) focuses on end-to-end communication and provides features such as reliability, congestion avoidance, and guaranteeing that packets will be delivered in the same order that they were sent. UDP (User Datagram protocol) is often adopted for IoT transport for performance reasons.

- **Application Layer**

  The application layer (Layers 5, 6, and 7 in OSI) covers application-level messaging. HTTP/S is an example of an application layer protocol that is widely adopted across the internet.

Although the TCP/IP and OSI models provide you with useful abstractions for discussing networking protocols and specific technologies that implement each protocol, some protocols don't fit neatly into these layered models and are impractical. For example, the Transport Layer Security (TLS) protocol that implements encryption to ensure privacy and data integrity of network traffic can be considered to operate across OSI layers 4, 5, and 6.

NETWORK ACCESS AND PHYSICAL LAYER IOT NETWORK TECHNOLOGIES

IoT network technologies to be aware of toward the bottom of the protocol stack include cellular, Wifi, and Ethernet, as well as more specialized solutions such as LPWAN, Bluetooth Low Energy (BLE), ZigBee, NFC, and RFID.

NB-IoT is becoming the standard for LPWAN networks, according to Gartner. This IoT for All article tells more about NB-IoT.

The following are network technologies with brief descriptions of each:

- **LPWAN**
  (Low Power Wide Area Network) is a category of technologies designed for low-power, long-range wireless communication. They are ideal for large-scale deployments of low-power IoT devices such as wireless sensors. LPWAN technologies include LoRa (LongRange physical layer protocol), Haystack, SigFox, LTE-M, and NB-IoT (Narrow-Band IoT).
- **Cellular**
  The LPWAN NB-IoT and LTE-M standards address low-power, low-cost IoT communication options using existing cellular networks. NB-IoT is the newest of

these standards and is focused on long-range communication between large numbers of primarily indoor devices. LTE-M and NB-IoT were developed specifically for IoT, however existing cellular technologies are also frequently adopted for long-range wireless communication. While this has included 2G (GSM) in legacy devices (and currently being phased out), CDMA (also being retired or phased out), it also includes 3G, which is rapidly being phased out with several network providers retiring all 3G devices. 4G is still active and will be until 5G becomes fully available and implemented.

- **Bluetooth Low Energy (BLE)**

BLE is a low-power version of the popular Bluetooth 2.4 GHz wireless communication protocol. It is designed for short-range (no more than 100 meters) communication, typically in a star configuration, with a single primary device that controls several secondary devices. Bluetooth operates across both layers 1 (PHY) and 2 (MAC) of the OSI model. BLE is best suited to devices that transmit low volumes of data in bursts. Devices are designed to sleep and save power when they are not transmitting data. Personal IoT devices such as wearable health and fitness trackers, often use BLE.

- **ZigBee**

  ZigBee operates on 2.4GHz wireless communication spectrum. It has a longer range than BLE by up to 100 meters. It also has a slightly lower data rate (250 kbps maximum compared to 270 kbps for BLE) than BLE. ZigBee is a mesh network protocol. Unlike BLE, not all devices can sleep between bursts. Much depends on their position in the mesh and whether they need to act as routers or controllers within the mesh. ZigBee was designed for building and home automation applications. Another closely related technology to ZigBee is Z-Wave, which is also based on IEEE 802.15.4. Z-Wave was designed for home automation. It has been proprietary technology, but was recently released as a public domain specification.

- **NFC**

  The near field communication (NFC) protocol is used for very small range communication (up to 4 cm), such as holding an NFC card or tag next to a reader. NFC is often used for payment systems, but also useful for check-in systems and smart labels in asset tracking.

- **RFID**

  RFID stands for Radio Frequency Identification. RFID tags store identifiers and data. The tags are attached to devices and read by an RFID reader. The typical range of RFID is less than a meter. RFID tags can be active, passive, or assisted passive. Passive tags are ideal for devices without batteries, as the ID is passively

read by the reader. Active tags periodically broadcast their ID, while assisted passive tags become active when RFID reader is present. **Dash7** is a communication protocol that uses active RFID that is designed to be used within Industrial IoT applications for secure long-range communication. Similar to NFC, a typical use case for RFID is tracking inventory items within retail and industrial IoT applications.

- **Wifi**

  Wifi is standard wireless networking based on IEEE 802.11a/b/g/n specifications. 802.11n offers the highest data throughput, but at the cost of high-power consumption, so IoT devices might only use 802.11b or g for power conservation reasons. Although wifi is adopted within many prototype and current generation IoT devices, as longer-range and lower-power solutions become more widely available, it is likely that wifi will be superseded by lower-power alternatives.

- **Ethernet**

  Widely deployed for wired connectivity within local area networks, Ethernet implements the IEEE 802.3 standard. Not all IoT devices need to be stationery wireless . For example, sensor units installed within a building automation system can use wired networking technologies like Ethernet. Power line communication (PLC), an alternative hard-wired solution, uses existing electrical wiring instead of dedicated network cables.

INTERNET LAYER IOT NETWORK TECHNOLOGIES

Internet layer technologies (OSI Layer 3) identify and route packets of data. Technologies commonly adopted for IoT are related to this layer, and include IPv6, 6LoWPAN, and RPL.

- **IPv6**

  At the Internet layer, devices are identified by IP addresses. IPv6 is typically used for IoT applications over legacy IPv4 addressing. IPv4 is limited to 32-bit addresses, which only provide around 4.3 billion addresses in total, which is less than the current number of IoT devices that are connected, while IPv6 uses 128 bits, and so provides 2^{128} addresses (around 3.4×10^{38} or 340 billion billion billion billion) addresses. In practice, not all IoT devices need public addresses. Of the tens of billions of devices expected to connect via the IoT over the next few years, many will be deployed in private networks that use private address ranges and only communicate out to other devices or services on external networks by using gateways.

- **6LoWPAN**

  The IPv6 Low Power Wireless Personal Area Network (6LoWPAN) standard allows IPv6 to be used over 802.15.4 wireless networks. 6LoWPAN is often used for wireless sensor networks, and the Thread protocol for home automation devices also runs over 6LoWPAN.

- **RPL**
  The Internet Layer also covers routing. IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) is designed for routing IPv6 traffic over low-power networks like those networks implemented over 6LoWPAN. RPL (pronounced “ripple”) is designed for routing packets within constrained networks such as wireless sensor networks, where not all devices are reachable at all times and there are high or unpredictable amounts of packet loss. RPL can compute the optimal path by building up a graph of the nodes in the network based on dynamic metrics and constraints like minimizing energy consumption or latency.

APPLICATION LAYER IOT NETWORK TECHNOLOGIES

HTTP and HTTPS are ubiquitous across internet applications, which is true also within IoT, with RESTful HTTP and HTTPS interfaces widely deployed. CoAP (Constrained Application Protocol) is like a lightweight HTTP that is often used in combination with 6LoWPAN over UDP. Messaging protocols like MQTT, AMQP, and XMPP are also frequently used within IoT applications:

- **MQTT**
  Message Queue Telemetry Transport (MQTT) is a publish/subscribe-based messaging protocol that was designed for use in low bandwidth situations, particularly for sensors and mobile devices on unreliable networks.
- **AMQP**
  Advanced Message Queuing Protocol (AMQP) is an open standard messaging protocol that is used for message-oriented middleware. Most notably, AMQP is implemented by RabbitMQ.
- **XMPP**
  The Extensible Messaging and Presence Protocol (XMPP) was originally designed for real-time human-to-human communication including instant messaging. This protocol has been adapted for machine-to-machine (M2M) communication to implement lightweight middleware and for routing XML data. XMPP is primarily used with smart appliances.

Your choice of technologies at this layer will depend on the specific application requirements of your IoT project. For example, for a budget home automation system that involves several sensors, MQTT would be a good choice as it is great for implementing messaging on devices without much storage or processing power because the protocol is simple and lightweight to implement.

IOT NETWORKING CONSIDERATIONS AND CHALLENGES

When you consider which networking technologies to adopt within your IoT application, be mindful of the following constraints:

- Range
- Bandwidth
- Power usage
- Intermittent connectivity
- Interoperability
- Security

Range

Networks can be described in terms of the distances over which data is typically transmitted by the IoT devices attached to the network:

- **PAN(PersonalAreaNetwork)**
  PAN is short-range, where distances can be measured in meters, such as a wearable fitness tracker device that communicates with an app on a cell phone over BLE.
- **LAN(LocalAreaNetwork)**
  LAN is short- to medium-range, where distances can be up to hundreds of meters, such as home automation or sensors that are installed within a factory production line that communicate over wifi with a gateway device that is installed within the same building.
- **MAN (Metropolitan Area Network)**
  MAN is long-range (city wide), where distances are measured up to a few kilometers, such as smart parking sensors installed throughout a city that are connected in a mesh network topology.
- **WAN (Wide Area Network)**
  WAN is long-range, where distances can be measured in kilometers, such as agricultural sensors that are installed across a large farm or ranch that are used to monitor micro-climate environmental conditions across the property.

Your network should retrieve data from the IoT devices and transmit to its intended destination. Select a network protocol that matches the range is required. For example, do not choose BLE for a WAN application to operate over a range of several kilometers. If transmitting data over the required range presents a challenge, consider edge computing. Edge computing analyzes data directly from the devices rather than from a distant data center or elsewhere.

Bandwidth

Bandwidth is the amount of data that can be transmitted per unit of time. It limits the rate at which data can be collected from IoT devices and transmitted upstream. Bandwidth is affected by many factors, which include:

- The volume of data each device gathers and transmits
- The number of devices deployed
- Whether data is being sent as a constant stream or in intermittent bursts, and if any peak periods are notable

The packet size of the networking protocol should match up with the volume of data typically transmitted. It is inefficient to send packets padded with empty data. In contrast, there are overheads in splitting larger chunks of data up across too many small packets. Data transmission rates are not always symmetrical (that is, upload rates might be slower than download rates). So, if there is two-way communication between devices, data transmission needs to be factored in. Wireless and cellular networks are traditionally low bandwidth, so consider whether a wireless technology is the right choice for high-volume applications.

Consider whether all raw data must be transmitted. A possible solution is to capture less data by sampling less frequently. Thus, you'll capture fewer variables and may filter data from the device to drop insignificant data. If you aggregate the data before you transmit it, you reduce the volume of data transmitted. But this process affects flexibility and granularity in the upstream analysis. Aggregation and bursting are not always suitable for time-sensitive or latency-sensitive data. All of these techniques increase the data processing and storage requirements for the IoT device.

Power usage

Transmitting data from a device consumes power. Transmitting data over long ranges requires more power than over a short range. You must consider the power source – such as a battery, solar cell, or capacitor – of a device and its total lifecycle. A long and enduring lifecycle will not only provide greater reliability but reduce operating cost. Steps may be taken to help achieve longer power supply lifecycles. For example, to prolong the battery life, you can put the device into sleep mode whenever it is idle. Another best practice is to model the energy consumption of the device under different loads and different network conditions to ensure that the device's power supply and storage capacity matches with the power that is required to transmit the necessary data by using the networking technologies that you adopted.

Intermittent connectivity

IoT devices aren't always connected. In some cases, devices are designed to connect periodically. However, sometimes an unreliable network might cause devices to drop off due to connectivity issues. Sometimes quality of service issues, such as dealing with interference or channel contention on a wireless network using a shared spectrum. Designs should incorporate intermittent connectivity and seek any available solutions to provide uninterrupted service, should that be a critical factor for IoT landscape design.

Interoperability

Devices work with other devices, equipment, systems, and technology; they are interoperable. With so many different devices connecting to the IoT, interoperability can be a challenge. Adopting standard protocols has been a traditional approach for maintaining interoperability on the Internet. Standards are agreed upon by industry participants and avoid multiple different designs and directions. With proper standards, and participants who agree to them, incompatibility issues, hence interoperability issues may be avoided.

However, for the IoT, standardization processes sometimes struggle to keep up with innovation and change. They are written and released based on upcoming versions of standards that are still subject to change. Consider the ecosystem around the technologies: Are they widely adopted? Are they open versus proprietary? How many implementations are available?

Using these questions to plan your IoT networks help plan better interoperability for a more robust IoT network.

Security

Security is a priority. Selection of networking technologies that implement end-to-end security, including authentication, encryption, and open port protection is crucial. IEEE 802.15.4 includes a security model that provides security features that include access control, message integrity, message confidentiality, and replay protection, which are implemented by technologies based on this standard such as ZigBee.

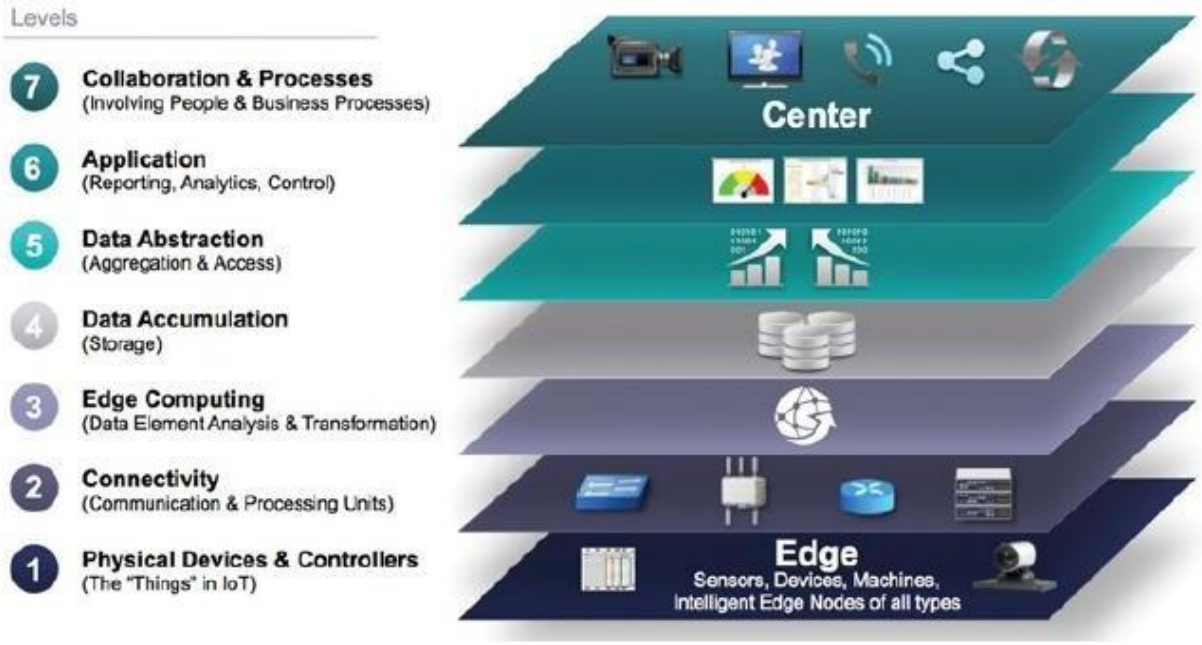
Consider the following factors in shaping a secure and safe IoT network:

- **Authentication**
  Adopt secure protocols to support authentication for devices, gateways, users, services, and applications. Consider using adopting the X.509 standard for device authentication.

- **Encryption**
  If you are using wifi, use Wireless Protected Access 2 (WPA2) for wireless network encryption. You may also adopt a Private Pre-Shared Key (PPSK) approach. To ensure privacy and data integrity for communication between applications, be sure to adopt TLS or Datagram Transport-Layer Security (DTLS), which is based on TLS, but adapted for unreliable connections that run over UDP. TLS encrypts application data and ensures its integrity.
- **Port protection**
  Port protection ensures that only the ports required for communication with the gateway or upstream applications or services remain open to external connections. All other ports should be disabled or protected by firewalls. Device ports might be exposed when exploiting Universal Plug and Play (UPnP) vulnerabilities. Thus, UPnP should be disabled on the router.

The IoT World Forum (IoTWF) Standardized Architecture

In 2014 the IoTWF architectural committee (led by Cisco, IBM, Rockwell Automation, and others)published a seven-layer IoT architectural reference model. While various IoT reference models exist, theone put forth by the IoT World Forum offers a clean, simplified perspective on IoT and includes edgecomputing, data storage, and access. It provides a succinct way of visualizing IoT from a technicalperspective. Each of the seven layers is broken down into specific functions, and security encompassesthe entire model. Figure belowdetails the IoT Reference Model published by the IoTWF.



As shown in Figure 2-2, the IoT Reference Model defines a set of levels with control flowing from thecenter (this could be either a cloud service or a dedicated data center), to the edge,

which includessensors, devices, machines, and other types of intelligent end nodes. In general, data travels up the stack,originating from the edge, and goes northbound to the center. Using this reference model, we are able toachieve the following:

- Decompose the IoT problem into smaller parts
- Identify different technologies at each layer and how they relate to one another
- Define a system in which different parts can be provided by different vendors
- Have a process of defining interfaces that leads to interoperability
- Define a tiered security model that is enforced at the transition points between levels

The following sections look more closely at each of the seven layers of the IoT Reference Model.

Layer 1: Physical Devices and Controllers Layer

The first layer of the IoT Reference Model is the physical devices and controllers layer. This layer is hometo the “things” in the Internet of Things, including the various endpoint devices and sensors that send andreceive information. The size of these “things” can range from almost microscopic sensors to giantmachines in a factory. Their primary function is generating data and being capable of being queriedand/or controlled over a network.

Layer 2: Connectivity Layer

In the second layer of the IoT Reference Model, the focus is on connectivity. The most important functionof this IoT layer is the reliable and timely transmission of data. More specifically, this includestransmissions between Layer 1 devices and the network and between the network and informationprocessing that occurs at Layer 3 (the edge computing layer).As you may notice, the connectivity layer encompasses all networking elements of IoT and doesn’t reallydistinguish between the last-mile network (the network between the sensor/endpoint and the IoT gateway,discussed later in this chapter), gateway, and backhaul networks. Functions of the connectivity layer aredetailed in Figure 2-3.

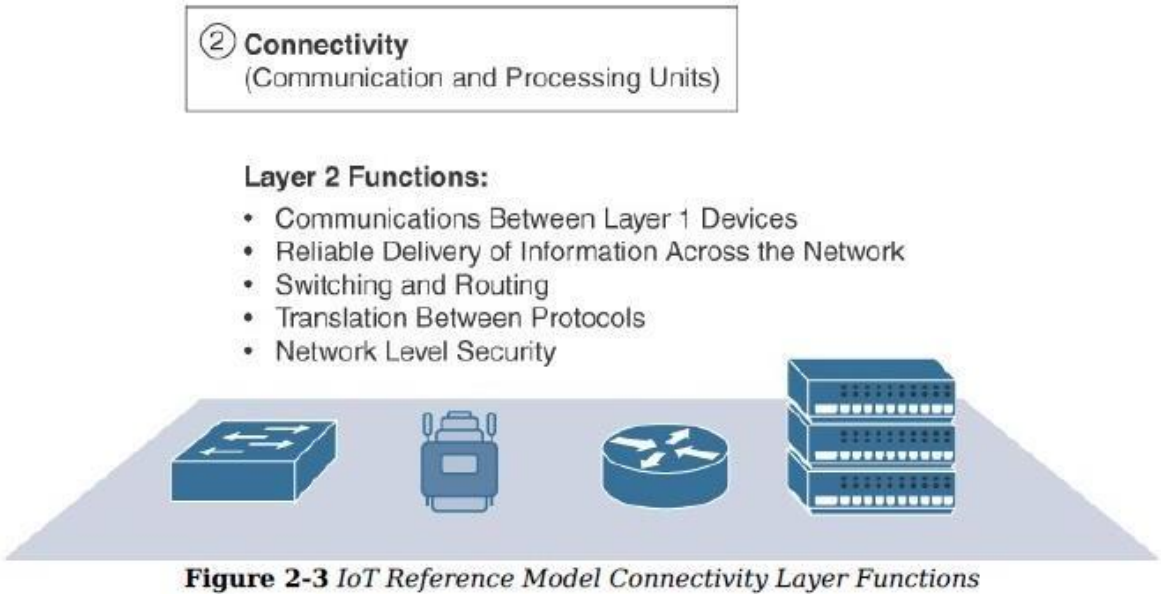


Figure 2-3 *IoT Reference Model Connectivity Layer Functions*

Layer 3: Edge Computing Layer

Edge computing is the role of Layer 3. Edge computing is often referred to as the "fog" layer and isdiscussed in the section "Fog Computing," later in this chapter. At this layer, the emphasis is on datareduction and converting network data flows into information that is ready for storage and processing byhigher layers. One of the basic principles of this reference model is that information processing is initiated

as early and as close to the edge of the network as possible. Figure 2-4 highlights the functions handledby Layer 3 of the IoT Reference Model.

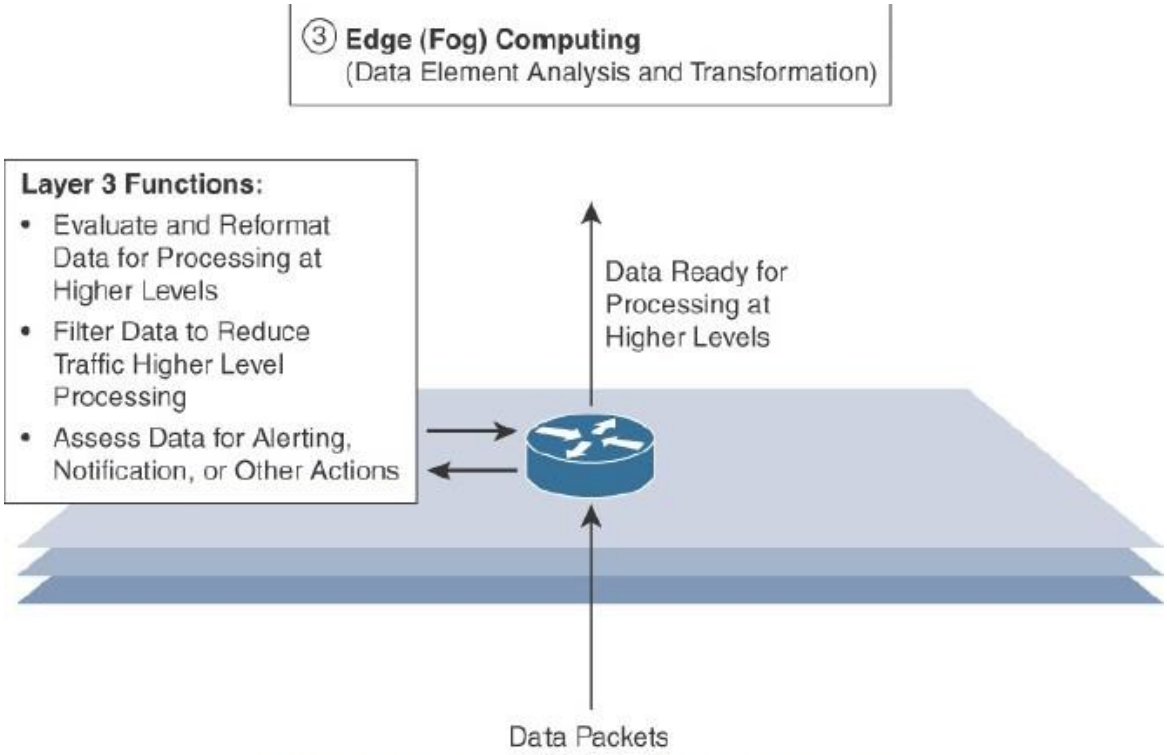


Figure 2-4 *IoT Reference Model Layer 3 Functions*

Another important function that occurs at Layer 3 is the evaluation of data to see if it can be filtered oraggregated before being sent to a higher layer. This also allows for data to be reformatted or decoded,making additional processing by other systems easier. Thus, a critical function is assessing the data to seeif predefined thresholds are crossed and any action or alerts need to be sent.

Upper Layers: Layers 4–7

The upper layers deal with handling and processing the IoT data generated by the bottom layer. For thesake of completeness, Layers 4–7 of the IoT Reference Model are summarized in Table 2-2.

IoT Reference Model Layer	Functions
Layer 4: Data accumulation layer	Captures data and stores it so it is usable by applications when necessary. Converts event-based data to query-based processing.
Layer 5: Data abstraction layer	Reconciles multiple data formats and ensures consistent semantics from various sources. Confirms that the data set is complete and consolidates data into one place or multiple data stores using virtualization.
Layer 6: Applications layer	Interprets data using software applications. Applications may monitor, control, and provide reports based on the analysis of the data.
Layer 7: Collaboration and processes layer	Consumes and shares the application information. Collaborating on and communicating IoT information often requires multiple steps, and it is what makes IoT useful. This layer can change business processes and delivers the benefits of IoT.

Table 2-2 *Summary of Layers 4–7 of the IoTWF Reference Model*

M2M Communication

Machine-to-machine communication, or M2M, is exactly as it sounds: two machines "communicating," or exchanging data, without human interfacing or interaction. This includes serial connection, powerline connection (PLC), or wireless communications in the industrial Internet of Things (IoT). Switching over to wireless has made M2M communication much easier and enabled more applications to be connected.

In general, when someone says M2M communication, they often are referring to cellular communication for embedded devices. Examples of M2M communication in this case would be vending machines sending out inventory information or ATM machines getting authorization to despense cash.

As businesses have realized the value of M2M, it has taken on a new name: the Internet of Things (IoT). IoT and M2M have similar promises: to fundamentally change the way the world operates. Just like IoT, M2M allows virtually any sensor to communicate, which opens up the possibility of systems monitoring themselves and automatically responding to changes in the environment, with a much reduced need for human involvement. M2M and IoT are almost synonymous—the exception is IoT (the newer term) typically refers to wireless communications, whereas M2M can refer to any two machines—wired or wireless—communicating with one another.

Traditionally, M2M focused on “industrial telematics,” which is a fancy way of explaining data transfer for some commercial benefit. But many original uses of M2M still stand today, like smart meters. Wireless M2M has been dominated by cellular since it came out in the mid-2000’s with 2G cell networks. Because of this, the cellular market has tried to brand M2M as an inherently cellular thing by offering M2M data plans. But cellular M2M is only one subsection of the market, and it shouldn’t be thought of as a cellular-only area.

How M2M Works

As previously stated, machine-to-machine communication makes the Internet of Things possible. According to Forbes, M2M is among the fastest-growing types of connected device technologies in the market right now, largely because M2M technologies can connect millions of devices within a single network. The range of connected devices includes anything from vending machines to medical equipment to vehicles to buildings. Virtually anything that houses sensor or control technology can be connected to some sort of wireless network.

This sounds complex, but the driving thought behind the idea is quite simple. Essentially, M2M networks are very similar to LAN or WAN networks, but are exclusively used to allow machines, sensors, and controls, to communicate. These devices feed information they collect back to other devices in the network. This process allows a human (or an intelligent control unit) to assess what is going on across the whole network and issue appropriate instructions to member devices.

M2M Applications

The possibilities in the realm of M2M can be seen in four major use cases, which we’ve detailed below:

1. MANUFACTURING

Every manufacturing environment—whether it’s food processing or general product manufacturing—relies on technology to ensure costs are managed properly and processes are executed efficiently. Automating manufacturing processes within such a fast-paced environment is expected to improve processes even more. In the manufacturing world, this could involve highly automated equipment maintenance and safety procedures.

For example, M2M tools allow business owners to be alerted on their smartphones when an important piece of equipment needs servicing, so they can address issues as quickly as they arise. Sophisticated networks of sensors connected to the Internet could even order replacement parts automatically.

2. HOME APPLIANCES

IoT already affects home appliance connectivity through platforms like Nest. However, M2M is expected to take home-based IoT to the next level. Manufacturers like LG and Samsung are already slowly unveiling smart home appliances to help ensure a higher quality of life for occupants.

For example, an M2M-capable washing machine could send alerts to the owners' smart devices once it finishes washing or drying, and a smart refrigerator could automatically order groceries from Amazon once its inventory is depleted. There are many more examples of home automation that can potentially improve quality of life for residents, including systems that allow members of the household to remotely control HVAC systems using their mobile devices. In situations where a homeowner decides to leave work early, he or she could contact the home heating system before leaving work to make sure the temperature at home will be comfortable upon arrival.

3. HEALTHCARE DEVICE MANAGEMENT

One of the biggest opportunities for M2M technology is in the realm of health care. With M2M technology, hospitals can automate processes to ensure the highest levels of treatment. Using devices that can react faster than a human healthcare professional in an emergency situation make this possible. For instance, when a patient's vital signs drop below normal, an M2M-connected life support device could automatically administer oxygen and additional care until a healthcare professional arrives on the scene. M2M also allows patients to be monitored in their own homes instead of in hospitals or care centers. For example, devices that track a frail or elderly person's normal movements can detect when he or she has had a fall and alert a healthcare worker to the situation.

4. SMART UTILITY MANAGEMENT

In the new age of energy efficiency, automation will quickly become the new normal. As energy companies look for new ways to automate the metering process, M2M comes to the rescue, helping energy companies automatically gather energy consumption data, so they can accurately bill customers. Smart meters can track how much energy a household or business uses and automatically alert the energy company, which supplants sending out an employee to read the meter or requiring the customer to provide a reading. This is even more important as utilities move toward more dynamic pricing models, charging consumers more for energy usage during peak times.

A few key analysts predict that soon, every object or device will need to be able to connect to the cloud. This is a bold but seemingly accurate statement. As more consumers, users, and business owners demand deeper connectivity, technology will need to be continually

equipped to meet the needs and challenges of tomorrow. This will empower a wide range of highly automated processes, from equipment repairs and firmware upgrades to system diagnostics, data retrieval, and analysis. Information will be delivered to users, engineers, data scientists, and key decision-makers in real time, and it will eliminate the need for guesswork.

The Value Of M2M

Growth in the M2M and IoT markets has been growing rapidly, and according to many reports, growth will continue. Strategy Analytics believes that low power, wide-area network (LPWAN) connections will grow from 11 million in 2014 to 5 billion in 2022. And IDC says the market for worldwide IoT solutions will go from $1.9 trillion in 2013 to $7.1 trillion in 2020.

Many big cell operators, like AT&T and Verizon, see this potential and are rolling out their own M2M platforms. Intel, PTC, and Wipro are are all marketing heavily in M2M and working to take advantage of this major industry growth spurt. But there is still a great opportunity for new technology companies to engage in highly automated solutions to help streamline processes in nearly any type of industry. We're certain we'll see a huge influx of companies who begin to innovate in this area in the next five years.

However, as the cost of M2M communication continues to decrease, companies must determine how they will create value for businesses and customers. In our mind, the opportunity and value for M2M doesn't lie in the more traditional layers of the communication world. Cell carriers and hardware manufacturers, for example, are beginning to look into full-stack offerings that enable M2M and IoT product development. We strongly believe value lies in the application side of things, and the growth in this industry will be driven by smart applications from this point forward.

Companies shouldn't think about IoT or M2M for the sake of IoT or M2M. Instead, they should focus on optimizing their business models or providing new value for their customers. For example, if you're a logistics company like FedEx or UPS, you have obvious choices for automated logistics decisions made by machines. But if you're a retailer, the transition to automation may not be as obvious. It's one thing to think of a "cool" automated process—say, creating advertising that is automatically tied to a specific customer through the use of M2M technology—but before you move forward with the process, you have to consider the value you're getting out of it. How much does it cost to implement? Will any company considering a move into the IoT space needs to understand what its business model is, how it will make money, and how it will provide value for customers or internal processes.

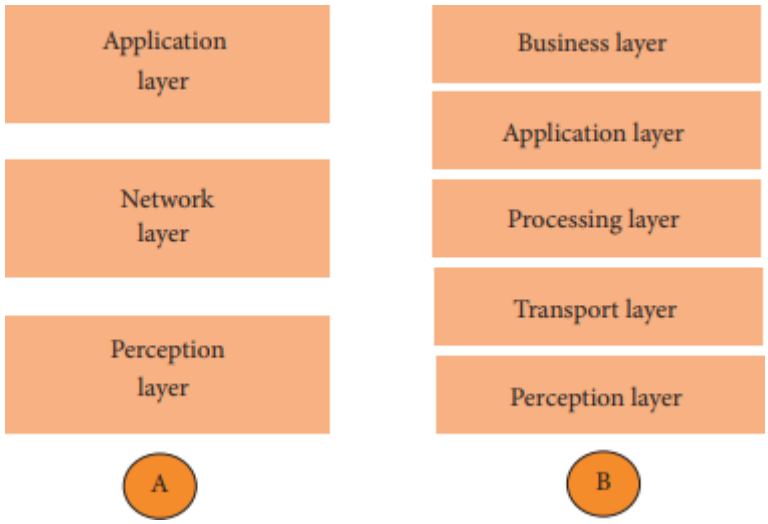
Architecture of IoT

Figure below has three layers, namely, the perception, network, and application layers.

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

(ii) The network layer is responsible for connecting to other smart things, network devices, and servers. Its features are also used for transmitting and processing sensor data.

(iii) The application layer is responsible for delivering application specific services to the user. It defines various applications in which the Internet of Things can be deployed, for example, smart homes, smart cities, and smart health.



The three-layer architecture defines the main idea of the Internet of Things, but it is not sufficient for research on IoT because research often focuses on finer aspects of the Internet of Things. That is why, we have many more layered architectures proposed in the literature. One is the fivelayer architecture, which additionally includes the processing and business layers [3–6]. The five layers are perception, transport, processing, application, and business layers (see Figure 1). The role of the perception and application layers is the same as the architecture with three layers. We outline the function of the remaining three layers.

(i) The transport layer transfers the sensor data from the perception layer to the processing layer and vice versa through networks such as wireless, 3G, LAN, Bluetooth, RFID, and NFC.

(ii) The processing layer is also known as the middleware layer. It stores, analyzes, and processes huge amounts of data that comes from the transport layer. It can manage and provide a diverse set of services to the

lower layers. It employs many technologies such as databases, cloud computing, and big data processing modules.

(iii) The business layer manages the whole IoT system, including applications, business and profit models, and users' privacy. The business layer is out of the scope of this paper. Hence, we do not discuss it further.

Core IoT Functional Stack

The IoT network must be designed to support its unique requirements and constraints. This section provides an overview of the full networking stack, from sensors all the way to the applications layer.

The Core IoT Functional Stack IoT networks are built around the concept of "things," or smart objects performing functions and delivering new connected services. These objects are "smart" because they use a combination of contextual information and configured goals to perform actions. These actions can be self-contained (that is, the smart object does not rely on external systems for its actions); however, in most cases, the "thing" interacts with an external system to report information that the smart object collects, to exchange with other objects, or to interact with a management platform. In this case, the management platform can be used to process data collected from the smart object and also guide the behavior of the smart object. From an architectural standpoint, several components have to work together for an IoT network to be operational: "Things" layer: At this layer, the physical devices need to fit the constraints of the environment in which they are deployed while still being able to provide the information needed. Communications network layer: When smart objects are not self-contained, they need to communicate with an external system. In many cases, this communication uses a wireless technology. This layer has four sublayers: Access network sublayer: The last mile of the IoT network is the access network. This is typically made up of wireless technologies such as 802.11ah, 802.15.4g, and LoRa. The sensors connected to the access network may also be wired. Gateways and backhaul network sublayer: A common communication system organizes multiple smart objects in a given area around a common gateway. The gateway communicates directly with the smart objects. The role of the gateway is to forward the collected information through a longer-range medium (called the backhaul) to a headend central station where the information is processed. This information exchange is a Layer 7 (application) function, which is the reason this object is called a gateway. On IP networks, this gateway also forwards packets from one IP network to another, and it therefore acts as a router. Network transport sublayer: For communication to be successful, network and transport layer protocols such as IP and UDP must be implemented to support the variety of devices to connect and media to use. IoT network management sublayer: Additional protocols must be in place to allow the headend applications to exchange data with the sensors. Examples include CoAP and MQTT.

Application and analytics layer: At the upper layer, an application needs to process the collected data, not only to control the smart objects when necessary, but to make intelligent decision based on the information collected and, in turn, instruct the "things" or other systems to adapt to the analyzed conditions and change their behaviors or parameters. The following sections examine these elements and help you architect your IoT communication network.

Layer 1: Things: Sensors and Actuators Layer

Most IoT networks start from the object, or "thing," that needs to be connected.From an architectural standpoint, the variety of smart object types, shapes, and needs drive the variety of IoT protocols and architectures. There are myriad ways to classify smart objects. One architectural classification could be:

- **Battery-powered or power-connected:** This classification is based on whether the object carries its own energy supply or receives continuous power from an external power source. Battery- powered things can be moved more easily than line-powered objects. However, batteries limit the lifetime and amount of energy that the object is allowed to consume, thus driving transmission range and frequency.
- **Mobile or static:** This classification is based on whether the "thing" should move or always stay at the same location. A sensor may be mobile because it is moved from one object to another (for example, a viscosity sensor moved from batch to batch in a chemical plant) or because it is attached to a moving object (for example, a location sensor on moving goods in a warehouse or factory floor). The frequency of the movement may also vary, from occasional to permanent. The range of mobility (from a few inches to miles away) often drives the possible power source.
- **Low or high reporting frequency:** This classification is based on how often the object should report monitored parameters. A rust sensor may report values once a month. A motion sensor may report acceleration several hundred times per second. Higher frequencies drive higher energy consumption, which may create constraints on the possible power source (and therefore the object mobility) and the transmission range.
- **Simple or rich data:** This classification is based on the quantity of data exchanged at each report cycle. A humidity sensor in a field may report a simple daily index value (on a binary scale from 0 to 255), while an engine sensor may report hundreds of parameters, from temperature to pressure, gas velocity, compression speed, carbon index, and many others. Richer data typically drives higher power consumption. This classification is often combined with the previous to determine the object data throughput (low throughput to high throughput). You may want to keep in mind that throughput is a combined metric. A medium-throughput object may send simple data at rather high frequency (in which case the flow structure looks

continuous), or may send rich data at rather low frequency (in which case the flow structure looks bursty).

- **Report range:** This classification is based on the distance at which the gateway is located. For example, for your fitness band to communicate with your phone, it needs to be located a few meters away at most. The assumption is that your phone needs to be at visual distance for you to consult the reported data on the phone screen. If the phone is far away, you typically do not use it, and reporting data from the band to the phone is not necessary. By contrast, a moisture sensor in the asphalt of a road may need to communicate with its reader several hundred meters or even kilometers away.
- **Object density per cell:** This classification is based on the number of smart objects (with a similar need to communicate) over a given area, connected to the same gateway. An oil pipeline may utilize a single sensor at key locations every few miles. By contrast, telescopes like the SETI Colossus telescope at the Whipple Observatory deploy hundreds, and sometimes thousands, of mirrors over a small area, each with multiple gyroscopes, gravity, and vibration sensors.

Layer 2: Communications Network Layer

Once you have determined the influence of the smart object form factor over its transmission capabilities (transmission range, data volume and frequency, sensor density and mobility), you are ready to connect the object and communicate. Compute and network assets used in IoT can be very different from those in IT environments. The difference in the physical form factors between devices used by IT and OT is obvious even to the most casual of observers. What typically drives this is the physical environment in which the devices are deployed. What may not be as inherently obvious, however, is their operational differences. The operational differences must be understood in order to apply the correct handling to secure the target assets. Temperature variances are an easily understood metric. The cause for the variance is easily attributed to external weather forces and internal operating conditions. Remote external locations, such as those associated with mineral extraction or pipeline equipment can span from the heat of the Arabian Gulf to the cold of the Alaskan North Slope. Controls near the furnaces of a steel mill obviously require heat tolerance, and controls for cold food storage require the opposite. In some cases, these controls must handle extreme fluctuations as well. These extremes can be seen within a single deployment. For example, portions of the Tehachapi, California, wind farms are located in the Mojave Desert, while others are at an altitude of 1800 m in the surrounding mountains. As you can imagine, the wide variance in temperature takes a special piece of hardware that is capable of withstanding such harsh environments. Humidity fluctuations can impact the long-term success of a system as well. Well heads residing in the delta of the Niger River

will see very different conditions from those in the middle of the Arabian Desert. In some conditions, the systems could be exposed to direct liquid contact such as may be found with outdoor wireless devices or marine condition deployments. Less obvious are the operating extremes related to kinetic forces. Shock and vibration needs vary based on the deployment scenario. In some cases, the focus is on low-amplitude but constant vibrations, as may be expected on a bushing-mounted manufacturing system. In other cases, it could be a sudden acceleration or deceleration, such as may be experienced in peak ground acceleration of an earthquake or an impact on a mobile system such as high-speed rail or heavy-duty earth moving equipment. Solid particulates can also impact the gear. Most IT environments must contend with dust build-up that can become highly concentrated due to the effect of cooling fans. In less-controlled IT environments, that phenomenon can be accelerated due to higher concentrations of particulates. A deterrent to particulate build-up is to use fanless cooling, which necessitates a higher surface area, as is the case with heat transfer fins. Hazardous location design may also cause corrosive impact to the equipment. Caustic materials can impact connections over which power or communications travel. Furthermore, they can result in reduced thermal efficiency by potentially coating the heat transfer surfaces. In some scenarios, the concern is not how the environment can impact the equipment but how the equipment can impact the environment. For example, in a scenario in which volatile gases may be present, spark suppression is a critical design criterion. There is another class of device differentiators related to the external connectivity of the device for mounting or industrial function. Device mounting is one obvious difference between OT and IT environments. While there are rack mount environments in some industrial spaces, they are more frequently found among IT type assets. Within industrial environments, many compute and communication assets are placed within an enclosed space, such as a control cabinet where they will be vertically mounted on a DIN (Deutsches Institut für Normung) rail inside. In other scenarios, the devices might be mounted horizontally directly on a wall or on a fence. In contrast to most IT-based systems, industrial compute systems often transmit their state or receive inputs from external devices through an alarm channel. These may drive an indicator light (stack lights) to display the status of a process element from afar. This same element can also receive inputs to initiate actions within the system itself. Power supplies in OT systems are also frequently different from those commonly seen on standard IT equipment. A wider range of power variations are common attributes of industrial compute components. DC power sources are also common in many environments. Given the criticality of many systems, it is often required that redundant power supplies be built into the device itself. Extraneous power supplies, especially those not inherently mounted, are frowned upon, given the potential for accidental unplugging. In some utility cases, the system must be able to handle brief power outages and still continue to operate.

- **Access Network Sublayer**

There is a direct relationship between the IoT network technology you choose and the type of connectivity topology this technology allows. Each technology was designed with a certain number of use cases in mind (what to connect, where to connect, how much data to transport at what interval and over what distance). These use cases determined the frequency band that was expected to be most suitable, the frame structure matching the expected data pattern (packet size and communication intervals), and the possible topologies that these use cases illustrate. As IoT continues to grow exponentially, you will encounter a wide variety of applications and special use cases. For each of them, an access technology will be required. IoT sometimes reuses existing access technologies whose characteristics match more or less closely the IoT use case requirements. Whereas some access technologies were developed specifically for IoT use cases, others were not. One key parameter determining the choice of access technology is the range between the smart object and the information collector. Figure 2-9 lists some access technologies you may encounter in the IoT world and the expected transmission distances.

PAN (personal area network): Scale of a few meters. This is the personal space around a person. A common wireless technology for this scale is Bluetooth.

HAN (home area network): Scale of a few tens of meters. At this scale, common wireless technologies for IoT include ZigBee and Bluetooth Low Energy (BLE).

NAN (neighborhood area network): Scale of a few hundreds of meters. The term NAN is often used to refer to a group of house units from which data is collected.

FAN (field area network): Scale of several tens of meters to several hundred meters. FAN typically refers to an outdoor area larger than a single group of house units. The FAN is often seen as “open space” (and therefore not secured and not controlled). A FAN is sometimes viewed as a group of NANs, but some verticals see the FAN as a group of HANs or a group of smaller outdoor cells. As you can see, FAN and NAN may sometimes be used interchangeably. In most cases, the vertical context is clear enough to determine the grouping hierarchy.

LAN (local area network): Scale of up to 100 m. This term is very common in networking, and it is therefore also commonly used in the IoT space when standard networking technologies (such as Ethernet or IEEE 802.11) are used. Other networking classifications, such as MAN (metropolitan area network, with a range of up to a few kilometers) and WAN (wide area network, with a range of more than a few kilometers), are also commonly used.

Network Transport Sublayer

The previous section describes a hierarchical communication architecture in which a series of smart objects report to a gateway that conveys the reported data over another medium and up to a central station. However, practical implementations are often flexible, with multiple transversal communication paths. For example, consider the case of IoT for the energy grid. Your house may have a meter that reports the energy consumption to a gateway over a wireless technology. Other houses in your neighborhood (NAN) make the same report, likely to one or several gateways. The data to be transported is small and the interval is large (for example, four times per hour), resulting in a low-mobility, lowthroughput type of data structure, with transmission distances up to a mile. Several technologies (such as 802.11ah, 802.15.4, or LPWA) can be used for this collection segment. Other neighborhoods may also connect the same way, thus forming a FAN.

IoT Network Management Sublayer

IP, TCP, and UDP bring connectivity to IoT networks. Upper-layer protocols need to take care of data transmission between the smart objects and other systems. Multiple protocols have been leveraged or created to solve IoT data communication problems. Some networks rely on a push model (that is, a sensor reports at a regular interval or based on a local trigger), whereas others rely on a pull model (that is, an application queries the sensor over the network), and multiple hybrid approaches are also possible.

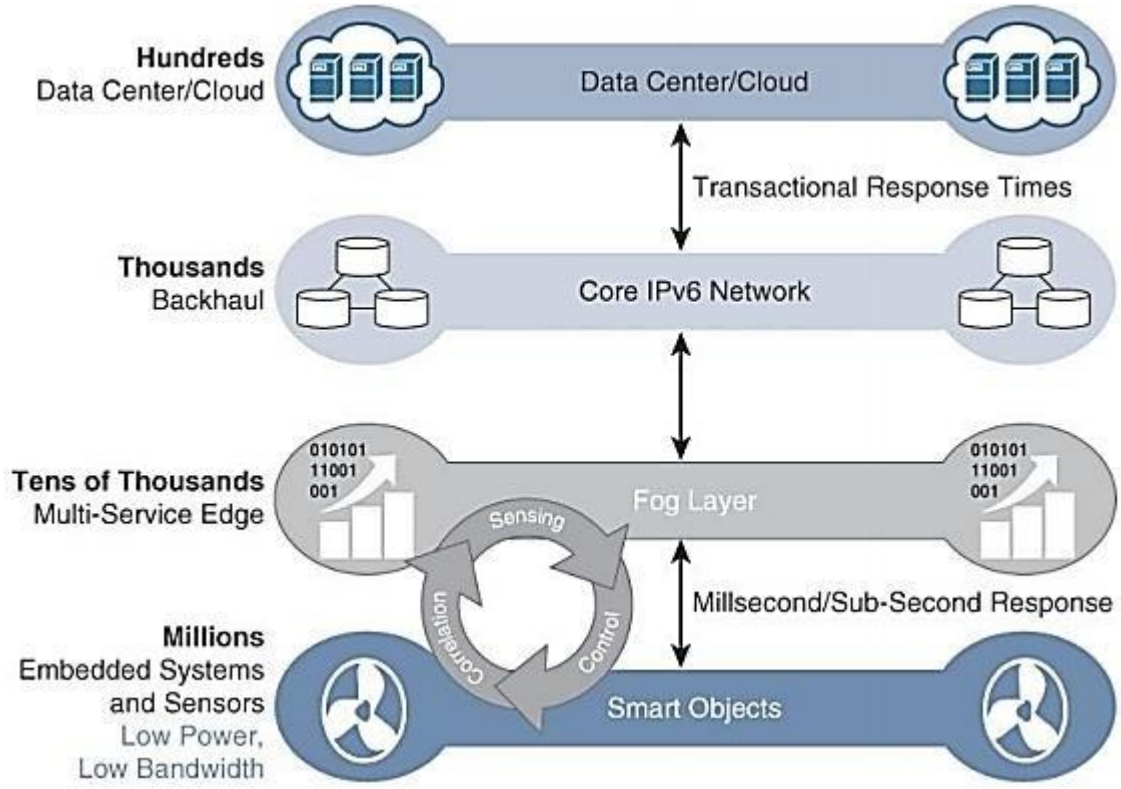
Layer 3: Applications and Analytics Layer

Once connected to a network, your smart objects exchange information with other systems. As soon as your IoT network spans more than a few sensors, the power of the Internet of Things appears in the applications that make use of the information exchanged with the smart objects. Analytics Versus Control Applications Multiple applications can help increase the efficiency of an IoT network. Each application collects data and provides a range of functions based on analyzing the collected data. It can be difficult to compare the features offered. Chapter 7, “Data and Analytics for IoT,” provides an in-depth analysis of the various application families. From an architectural standpoint, one basic classification can be as follows: Analytics application: This type of application collects data from multiple smart objects, processes the collected data, and displays information resulting from the data that was processed. The display can be about any aspect of the IoT network, from historical reports, statistics, or trends to individual system states. The important aspect is that the application processes the data to convey a view of the network that cannot be obtained from solely looking at the information displayed by a single

smart object. Control application: This type of application controls the behavior of the smart object or the behavior of an object related to the smart object. For example, a pressure sensor may be connected to a pump. A control application increases the pump speed when the connected sensor detects a drop in pressure. Control applications are very useful for controlling complex aspects of an IoT network with a logic that cannot be programmed inside a single IoT object, either because the configured changes are too complex to fit into the local system or because the configured changes rely on parameters that include elements outside the IoT object.

Fog Computing

The solution to the challenges mentioned in the previous section is to distribute data management throughout the IoT system, as close to the edge of the IP network as possible. The best-known embodiment of edge services in IoT is fog computing. Any device with computing, storage, and network connectivity can be a fog node. Examples include industrial controllers, switches, routers, embedded servers, and IoT gateways. Analyzing IoT data close to where it is collected minimizes latency, offloads gigabytes of network traffic from the core network, and keeps sensitive data inside the local network.



Fog services are typically accomplished very close to the edge device, sitting as close to the IoT endpoints as possible. One significant advantage of this is that the fog

node has contextual awareness of the sensors it is managing because of its geographic proximity to those sensors. For example, there might be a fog router on an oil derrick that is monitoring all the sensor activity at that location. Because the fog node is able to analyze information from all the sensors on that derrick, it can provide contextual analysis of the messages it is receiving and may decide to send back only the relevant information over the backhaul network to the cloud. In this way, it is performing distributed analytics such that the volume of data sent upstream is greatly reduced and is much more useful to application and analytics servers residing in the cloud.

The defining characteristic of fog computing are as follows:

Contextual location awareness and low latency: The fog node sits as close to the IoT endpoint as possible to deliver distributed computing.

Geographic distribution: In sharp contrast to the more centralized cloud, the services and applications targeted by the fog nodes demand widely distributed deployments.

Deployment near IoT endpoints: Fog nodes are typically deployed in the presence of a large number of IoT endpoints. For example, typical metering deployments often see 3000 to 4000 nodes per gateway router, which also functions as the fog computing node.

Wireless communication between the fog and the IoT endpoint: Although it is possible to connect wired nodes, the advantages of fog are greatest when dealing with a large number of endpoints, and wireless access is the easiest way to achieve such scale.

Use for real-time interactions: Important fog applications involve real-time interactions rather than batch processing. Preprocessing of data in the fog nodes allows upper-layer applications to perform batch processing on a subset of the data.

Edge Computing

Fog computing solutions are being adopted by many industries, and efforts to develop distributed applications and analytics tools are being introduced at an accelerating pace. The natural place for a fog node is in the network device that sits closest to the IoT endpoints, and these nodes are typically spread throughout an IoT network. However, in recent years, the concept of IoT computing has been pushed even further to the edge, and in some cases it now resides directly in the sensors and IoT devices.

Functional blocks of an IoT ecosystem

IoTdon’t exist in a void. A lone sensor isn’t really good for anything, nor is a bunch of them, for that matter, unless they are all connected to one another and to platforms that generate data for further use. This is what we call an **Internet of Things (IoT) ecosystem – a broad network of connected and interdependent devices and technologies that are applied by specialists towards a specific goal**, such as the creation of a smart city.
Obviously, there are limitless applications to the IoT and therefore we can speak of endless coexisting IoT ecosystems. But if you boil what is happening in the ecosystem down to the bare essentials, you will come up with a simple schema: a **device** collects data and sends it across the **network** to a **platform** that aggregates the data for future use by the **agent**. And so we have the key components to an IoT ecosystem: devices, networks, platforms, and agents. Let’s discuss them in more detail.

Four things form basic building blocks of the IoT system –sensors, processors, gateways, applications. Each of these nodes has to have its own characteristics in order to form an useful IoT system.

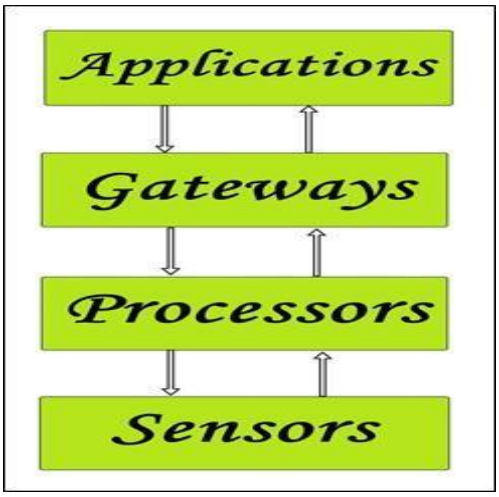


Figure 1: Simplified block diagram of the basic building blocks of the **IoT**

Sensors:

- These form the front end of the IoT devices. These are the so-called "Things" of the system. Their main purpose is to collect data from its surroundings (sensors) or give out data to its surrounding (actuators).
- These have to be uniquely identifiable devices with a unique IP address so that they can be easily identifiable over a large network.
- These have to be active in nature which means that they should be able to collect real-time data. These can either work on their own (autonomous in nature) or can be made to work by the user depending on their needs (user-controlled).
- Examples of sensors are gas sensor, water quality sensor, moisture sensor, etc.

Processors:

- Processors are the brain of the IoT system. Their main function is to process the data captured by the sensors and process them so as to extract the valuable data from the enormous amount of raw data collected. In a word, we can say that it gives intelligence to the data.
- Processors mostly work on real-time basis and can be easily controlled by applications. These are also responsible for securing the data – that is performing encryption and decryption of data.
- Embedded hardware devices, microcontroller, etc are the ones that process the data because they have processors attached to it.

Gateways:

- Gateways are responsible for routing the processed data and send it to proper locations for its (data) proper utilization.
- In other words, we can say that gateway helps in to and fro communication of the data. It provides network connectivity to the data. Network connectivity is essential for any IoT system to communicate.
- LAN, WAN, PAN, etc are examples of network gateways.

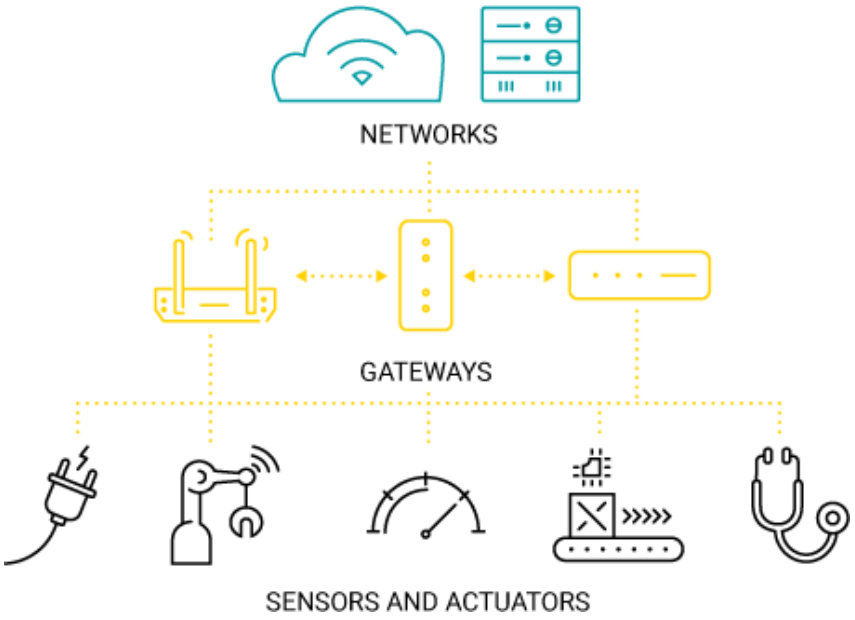
Applications:

- Applications form another end of an IoT system. Applications are essential for proper utilization of all the data collected.
- These cloud-based applications which are responsible for rendering the effective meaning to the data collected. Applications are controlled by users and are a delivery point of particular services.
- Examples of applications are home automation apps, security systems, industrial control hub, etc.

IoT devices

As we said earlier, there are many scenarios in which IoT can be employed and they all require different devices. Here, at the most basic level, we can speak of **sensors** (i.e. devices that sense things, such as temperature, motion, particles, etc.) and **actuators** (i.e. devices that act on things, such as switches or rotors).
Rarely, though, will a smart solution make do with just one type of an IoT sensor or an actuator. If you think of a smart surgical robot, for example, it will require hundreds, if not thousands, of components that measure different parameters and act accordingly. But even apparently less complicated solutions aren't truly that easy. Consider running a smart farm – for a plant to grow, it's not just a matter of measuring the humidity of the soil, but also its fertility; it's also a matter of providing proper irrigation based on insolation, and much more. So you need not just one, but many sensors and actuators that all have to work together.

When speaking of devices essential for the IoT ecosystem, one cannot forget about IoT gateways. They are a piece of hardware that is capable of "translating" and facilitating the essential connection between devices or between devices and the network and work as a kind of relay for the two. Which brings us to the next element of our puzzle…

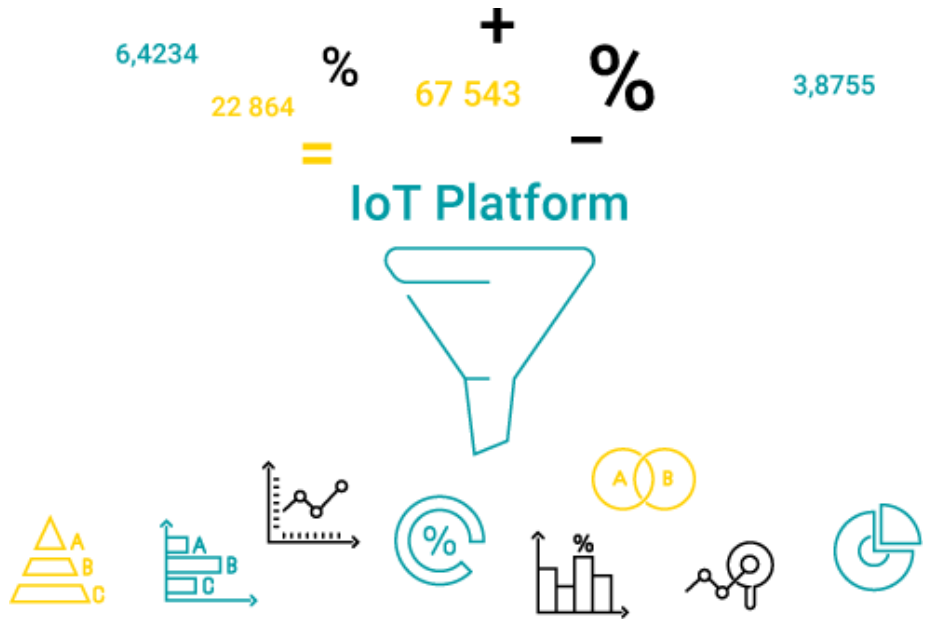


Networks

Based on what you read before, you may think: “Well, if an automatic door senses my presence and opens itself, is that IoT?” Obviously, it is not, because while that door has sensors and actuators, it is not connected to much else. And, as the name suggests, **the Internet of Things requires both things and the Internet (although there are cases of data delivery without the use of the Internet Protocol). Arguably, the real power of this concept lies in the connectivity.**
Again, based on your deployment needs, there are plenty of different IoT connectivity options, starting with the “classics,” such as WiFi or Bluetooth, to more specialized and field-oriented technologies, such as Low-Power Wide Area Networks (LPWAN). **They all differ in range and speed of data transfer, making them more or less appropriate for particular deployments.** Consider, for example, smart cars that require both high data speed and long range and juxtapose them with the smart farms we’ve mentioned that don’t necessarily need either.

IoT platform

Whether they are in the cloud or not, **IoT platforms are always the binder for any IoT ecosystem.** They are the quiet administrators that take care of device lifecycle management, so that you don’t have to worry about them. They are also the hub that collects and aggregates the data, allowing you to make sense of it. With the variety of platforms offered on the market and the breadth of claims their providers make, **the choice of the “ideal” IoT platform for a deployment is arguably the most significant, yet also the most difficult to make.** It shouldn’t be taken lightly, as it determines whether the IoT ecosystem will thrive or wither into oblivion.



The right IoT device management platform should be versatile and adaptable, as the IoT world is very fragmented and constantly shifting and you don't want the core element of your ecosystem to become the stumbling block of your deployment. **It should also be scalable,** so that your ecosystem can grow naturally, and **secure**, so it can do so without any threats.

Agents

Agents are all the people whose actions affect the IoT ecosystem. These may be the engineers who devise IoT deployments and design the platforms, it can also be the platform operators. But probably, most importantly, it's the stakeholders, who ultimately reap the results. After all, IoT deployments aren't just art for art's sake. **These complex ecosystems are put in place for a reason: to drive efficiency and improve the quality of life. And it is the agents who decide on how to use the devices, networks and platforms to achieve these results.** This is where technology and business converge, because it's business goals that very much shape the IoT ecosystem.

People are an essential part of this equation. Ecosystems are created by us, managed by us and, ultimately, it is our responsibility to realize their full potential. It is the devices that collect the data, but it is the people that make sense of it and put it to use. Similarly with networks and platforms, which are a necessary component of the ecosystem, but wouldn't be of much value if it weren't for the people who create and perfect them to fit their needs.

As was said, **an IoT ecosystem is a very complex concept that eludes easy classification, as its characteristics vary from deployment to deployment.** Much like our world, the IoT world comprises numerous different ecosystems that evolve and adapt. What they have in common is the idea and the people that make them happen: device manufacturers, service providers, application developers, and enterprises. **Yet in this ever changing landscape there still remains a lot of variety – the technology, represented by the devices, networks and platforms, always gets better.** This is particularly worth remembering, because the one mistake the inhabitants of an IoT ecosystem should never make, is to take it for granted. There is nothing more toxic for that landscape than stagnation and lock-in, so you should always be on the lookout for newer, better technologies that will help you flourish.

IoT ACCESS TECHNOLOGIES

Introduction:

This unit provides an in depth look at some of access technologies that are considered for connection of smart Objects. The most promising access technologies which are going forward in the IoT marketplace are discussed. The technologies highlighted here are the ones that are seen as having market and/or mind share. The following are the technologies for connecting smart objects:

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

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

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

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

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

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

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

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

IEEE 802.15.4

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

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

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

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

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

Standardization and Alliances:

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

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

The IEEE 802.15.4 PHY and MAC layers are the foundations for several networking protocol stacks. These protocol stacks make use of 802.15.4 at the physical and link layer levels, but the upper layers are different. These protocol stacks are promoted separately through various organizations and often commercialized. Some of the most well-known protocol stacks based on 802.15.4 are highlighted in Table 2-1.

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

ZigBee

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

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

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

the utility provider and can help coordinate usage between homes and businesses and the utility provider itself to provide more efficient operations.
The traditional ZigBee stack is illustrated in Figure 2.1. As mentioned previously, ZigBee utilizes the IEEE
802.15.4 Standard at the lower PHY and MAC layers. ZigBee specifies the network and security layer and application support layer that sit on top of the lower layers.

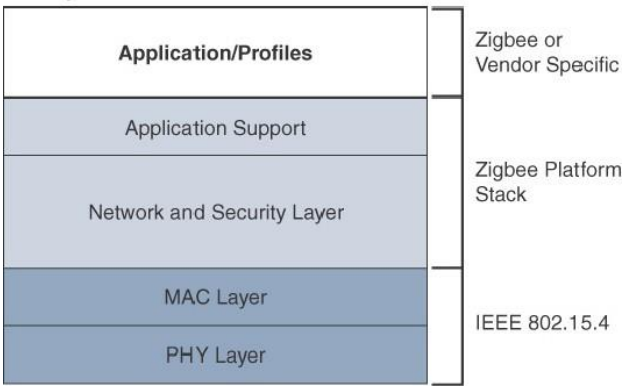


Figure 2.1 High Level ZigBee Protocol Stack

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

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

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

ZigBee IP

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

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

ZigBee IP is a critical part of the Smart Energy (SE) Profile 2.0 specification from the ZigBee Alliance. SE 2.0 is aimed at smart metering and residential energy management systems. In fact, ZigBee IP was designed specifically for SE 2.0 but it is not limited to this use case. Any other applications that need a standards based IoT stack can utilize Zigbee IP. The ZigBee IP stack is shown in Figure 2.2

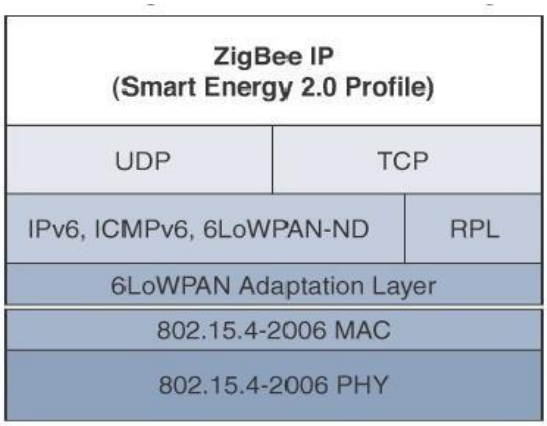


Figure 2.2 ZigBee IP protocol stack

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

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

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

Physical Layer

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

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

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

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

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

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

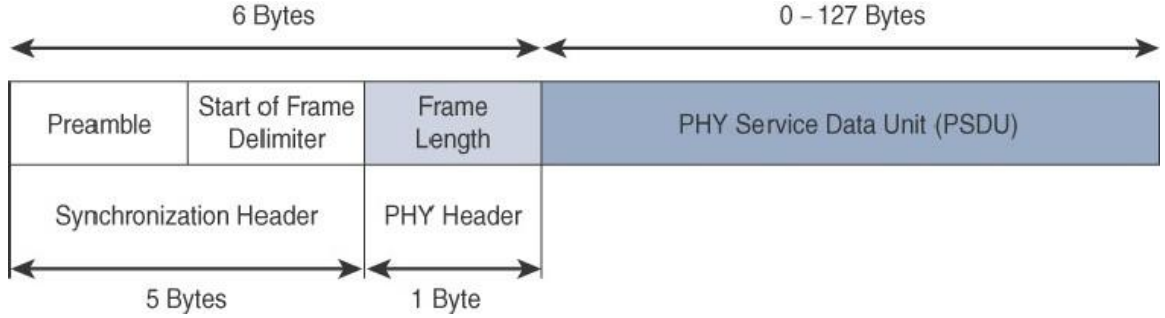


Figure 2.3 IEEE 802.15.4 PHY Format

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

MAC Layer

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

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

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

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

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

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

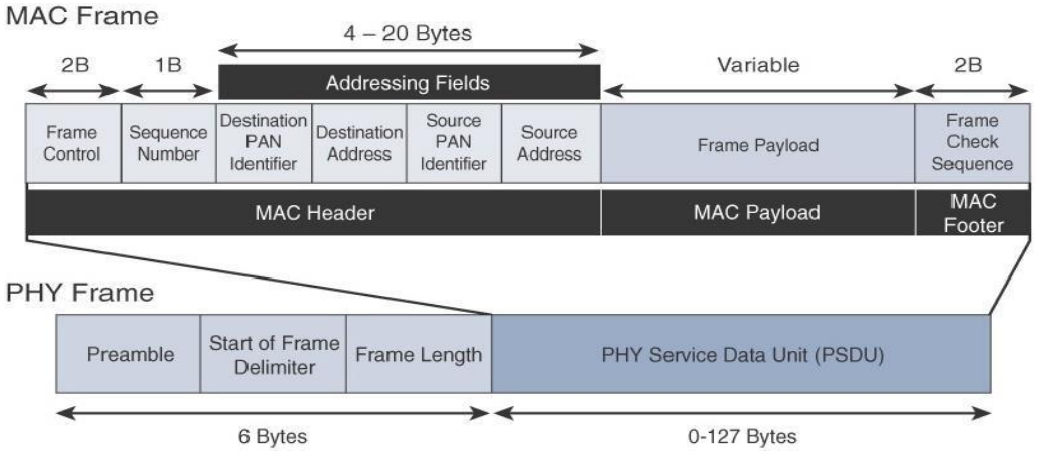


Figure 2.4 IEEE 802.15.4 MAC Format

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

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

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

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

Topology

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

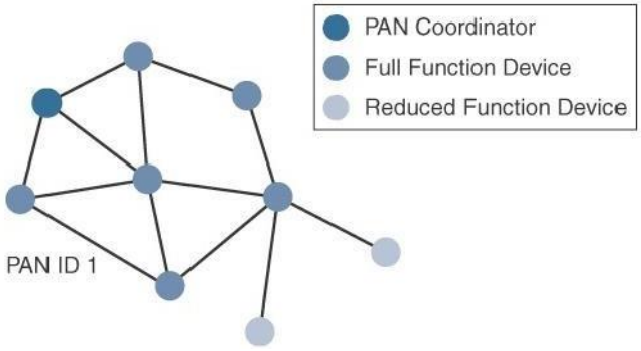


Figure 2.5: Sample Mesh Network Topology

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

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

Security

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

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

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

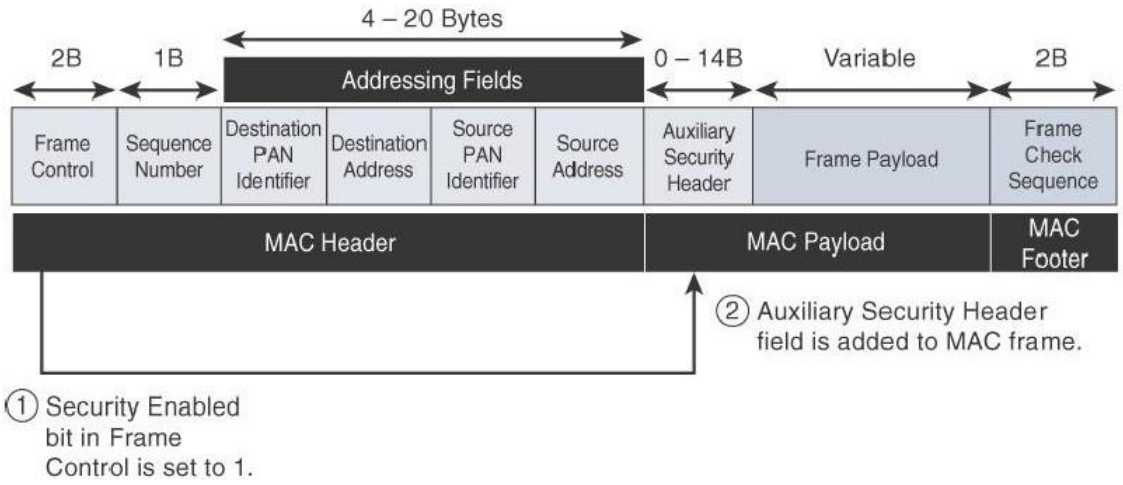


Figure 2.6: Frame format with the Auxiliary Security Header Field for 802.15.4-2006

Competitive Technologies

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

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

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

IEEE 802.15.4 Summary

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

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

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

IEEE 802.11ah

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

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

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

Standardization and Alliances

In July 2010, the IEEE 802.11 working group decided to work on an "industrial Wi-Fi" and created the IEEE 802.11ah group. The 802.11ah specification would operate in unlicensed sub-GHz frequency bands, similar to IEEE 802.15.4 and other LPWA technologies.

The industry organization that promotes Wi-Fi certifications and interoperability for 2.4 GHz and 5 GHz products is the Wi-Fi Alliance. The Wi-Fi Alliance is a similar body to the Wi-SUN Alliance.

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

Physical Layer

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

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

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

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

MAC Layer

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

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

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

Topology

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

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

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

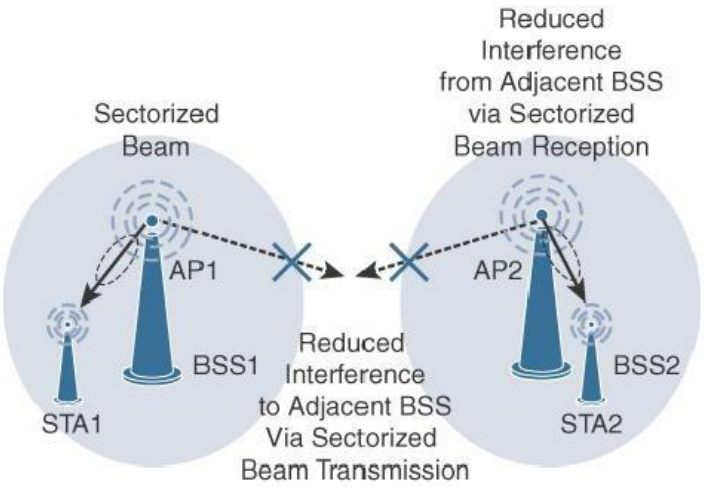


Figure 2.7: IEEE 802.11 ah Sectorization

Security

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

Competitive Technologies

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

IEEE 802.11ah Conclusions

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

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

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

LoRaWAN

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

Standardization and Alliances

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

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

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

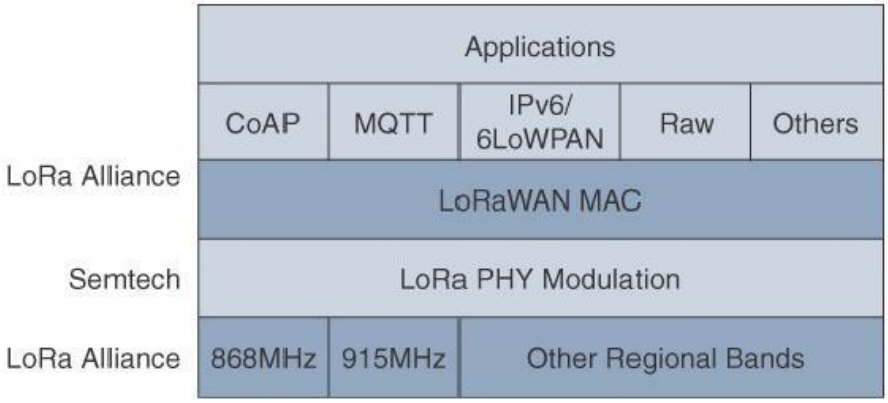


Figure 2.8 LoRa WAN Layers

Physical Layer

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

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

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

The data rate in LoRaWAN varies depending on the frequency bands and adaptive data rate (ADR). ADR is an algorithm that manages the data rate and radio signal for each endpoint. The ADR algorithm ensures that packets are delivered at the best data rate possible and that network performance is both optimal and scalable. Endpoints close to the gateways with good signal values transmit with the highest data rate, which enables a shorter transmission time over the wireless network, and the lowest transmit power. Meanwhile, endpoints at the edge of the link budget communicate at the lowest data rate and highest transmit power.

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

Table 2.2 LoRa WAN Data Rate example

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

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

MAC Layer

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

- **Class A:** This class is the default implementation. Optimized for battery-powered nodes, it allows bidirectional communications, where a given node is able to receive downstream traffic after transmitting. Two receive windows are available after each transmission.
- **Class B:** This class was designated “experimental” in LoRaWAN 1.0.1 until it can be better defined. A Class B node or endpoint should get additional receive windows compared to Class A, but gateways must be synchronized through a beaconing process.
- **Class C:** This class is particularly adapted for powered nodes. This classification enables a node to be continuously listening by keeping it’s receive window open when not transmitting.

LoRaWAN messages, either uplink or downlink, have a PHY payload composed of a 1-byte MAC header, a variable-byte MAC payload, and a MIC that is 4 bytes in length. The MAC payload size depends on the frequency band and the data rate, ranging from 59 to 230

bytes for the 863–870 MHz band and 19 to 250 bytes for the 902–928 MHz band. Figure 2.9 shows a high-level LoRaWAN MAC frame format.

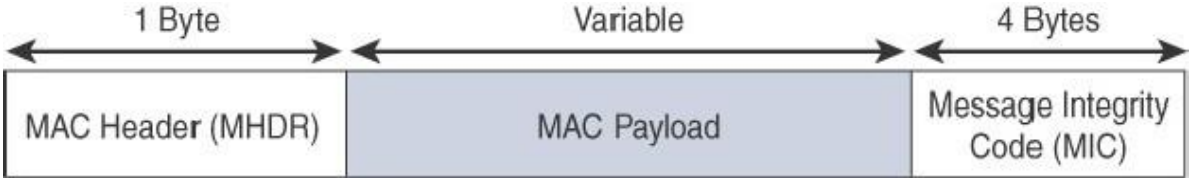


Figure 2.9 High- Level LoRaWAN MAC Frame Format

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

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

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

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

Topology

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

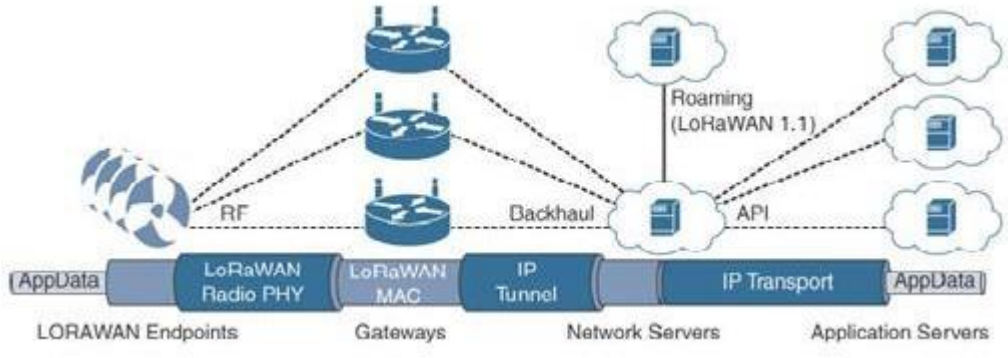


Figure 2.10 LoRa WAN Architecture

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

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

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

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

Security

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

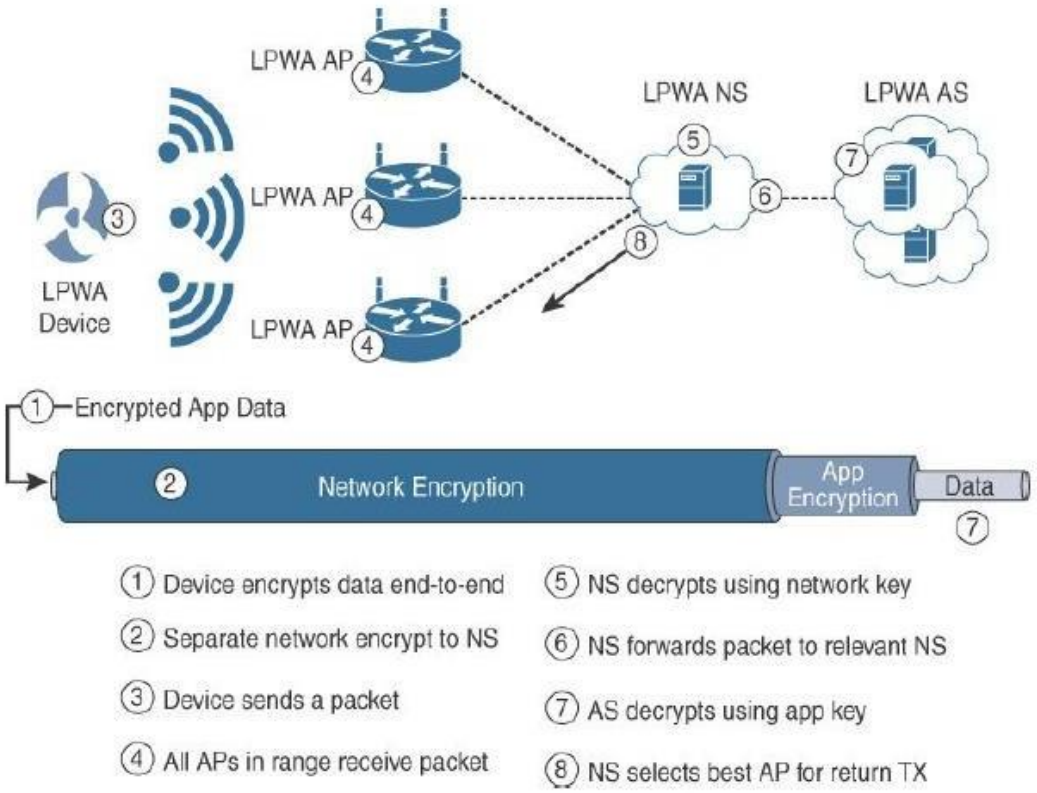


Figure 2.11 LoRaWAN Security

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

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

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

- **Activation by personalization (ABP):** Endpoints don't need to run a join procedure as their individual details, including DevAddr and the NwkSKey and AppSKey session keys, are preconfigured and stored in the end device. This same information is registered in the LoRaWAN network server.
- **Over-the-air activation (OTAA):** Endpoints are allowed to dynamically join a particular LoRaWAN network after successfully going through a join procedure. The join procedure must be done every time a session context is renewed. During the join process, which involves the sending and receiving of MAC layer join request and join accept messages, the node establishes its credentials with a LoRaWAN network server, exchanging its globally unique DevEUI, AppEUI, and AppKey. The AppKey is then used to derive the session NwkSKey and AppSKey keys.

Competitive Technologies

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

Table 2.3 Unlicensed LPWA Technology Comparison

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

THE NETWORK LAYER

Constrained Nodes

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

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

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

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

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

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

The definition of constrained nodes is evolving. The costs of computing power, memory, storage resources, and power consumption are generally decreasing. At the same time,

networking technologies continue to improve and offer more bandwidth and reliability. In the future, the push to optimize IP for constrained nodes will lessen as technology improvements and cost decreases address many of these challenges.

Constrained Networks

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

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

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

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

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

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

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

IP Versions

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

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

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

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

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

6LoWPAN

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

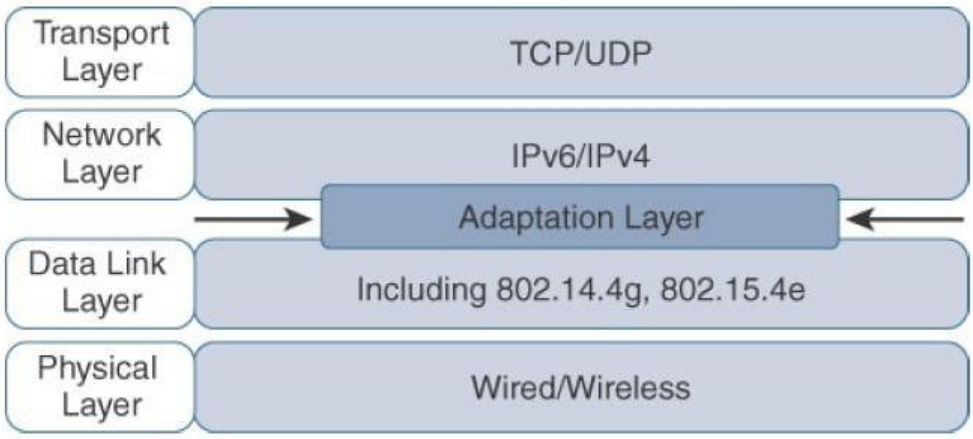


Figure 2.12: Optimizing IP for IoT Using an Adaptation Layer

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

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

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

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

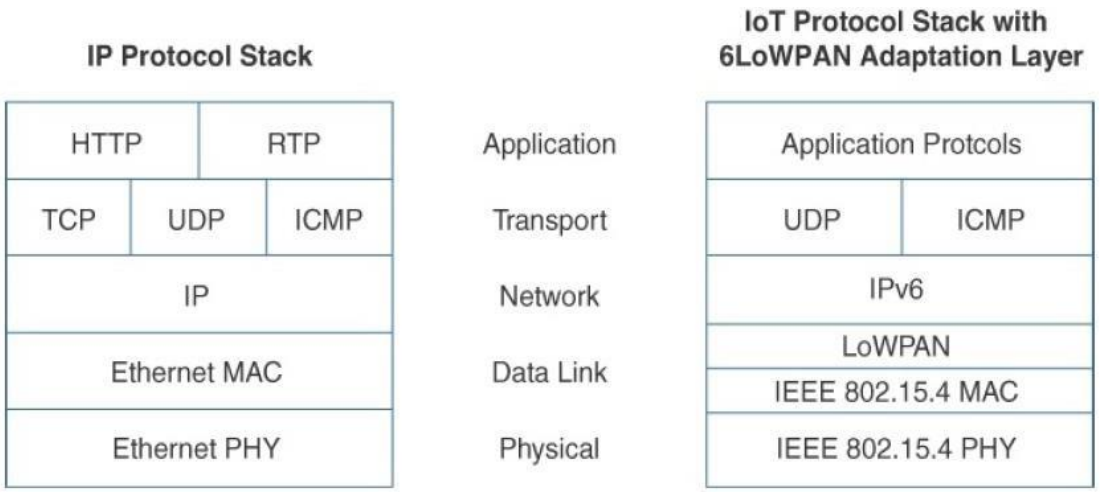


Figure 2.13: Comparison of an IoT Protocol Stack Utilizing 6LoWPAN and an IP Protocol Stack

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

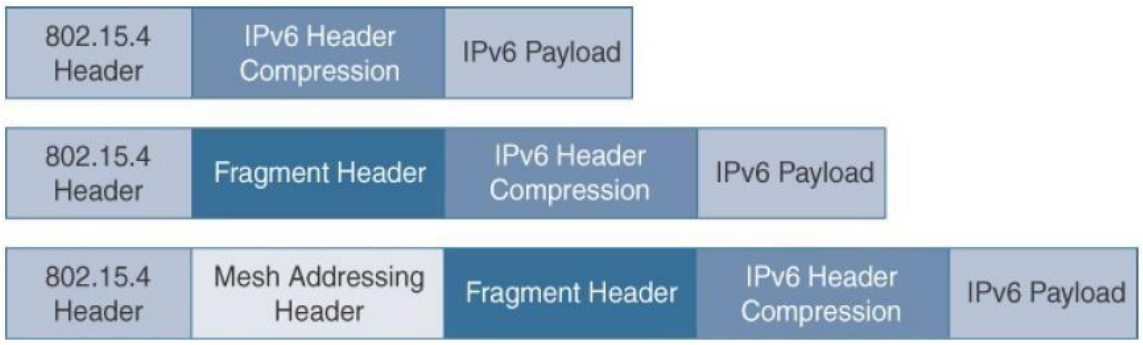


Figure 2.14 6LoWPAN Header Stack

Header Compression

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

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

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

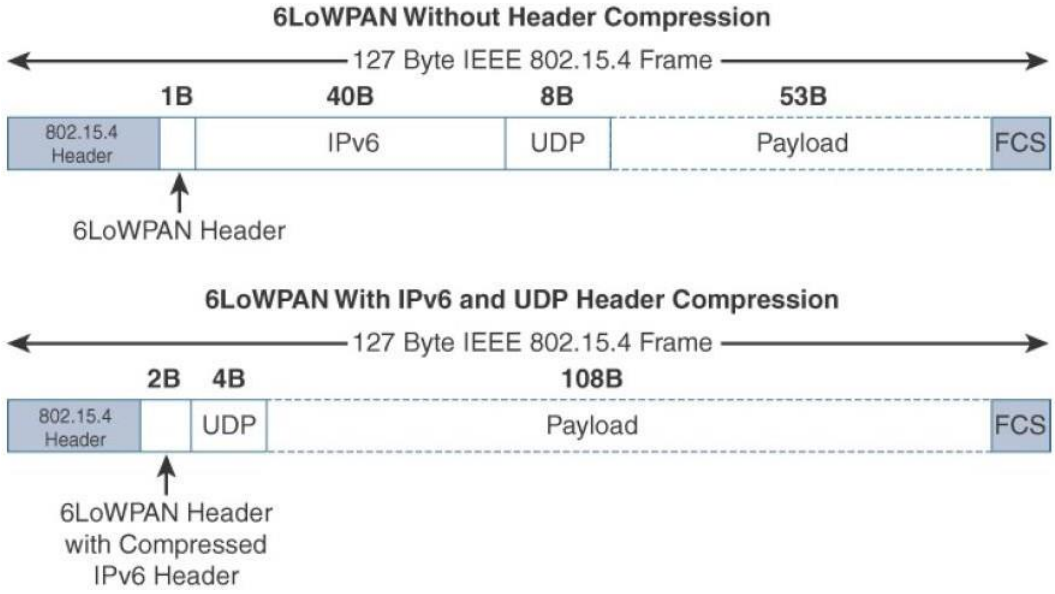


Figure 2.15 6LoWPAN Header Compression

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

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

Fragmentation

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

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

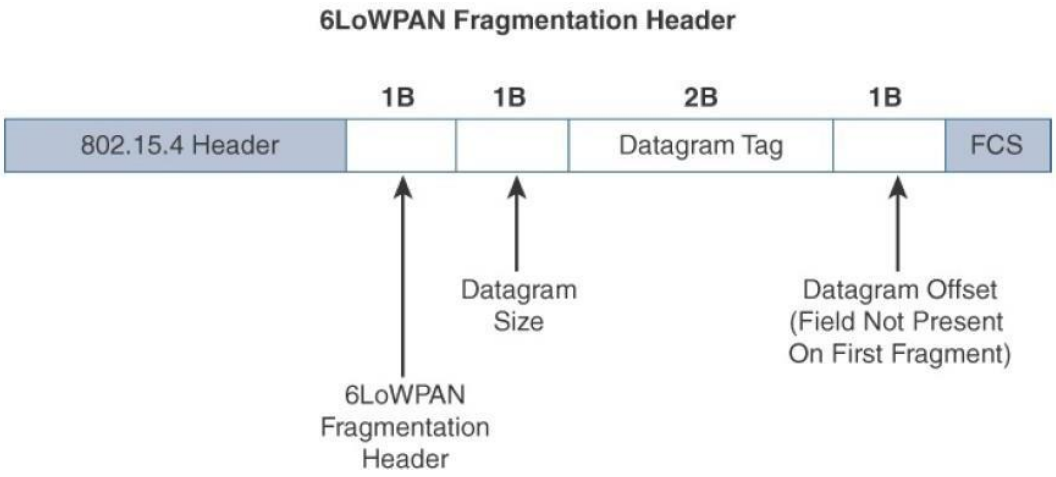


Figure 2.16 6LoWPAN Fragmentation Header

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

Mesh Addressing

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

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

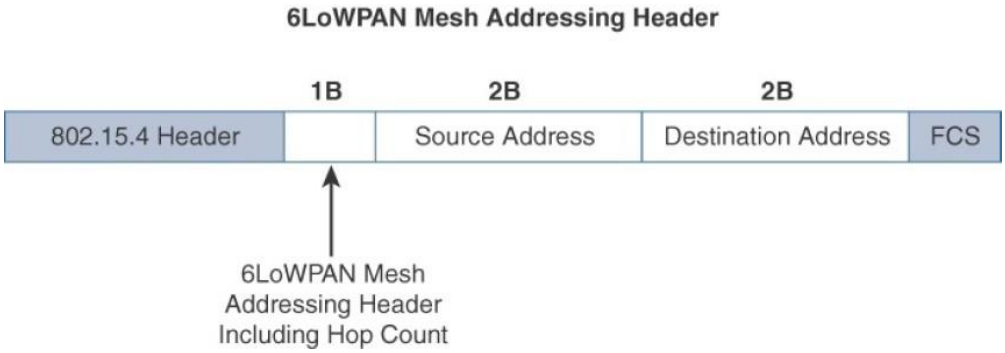


Figure 2.17: 6LoWPAN Mesh Addressing Header

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

Application Transport methods

SCADA

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

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

A Little Background on SCADA

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

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

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

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

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

Adapting SCADA for IP

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

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

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

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

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

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

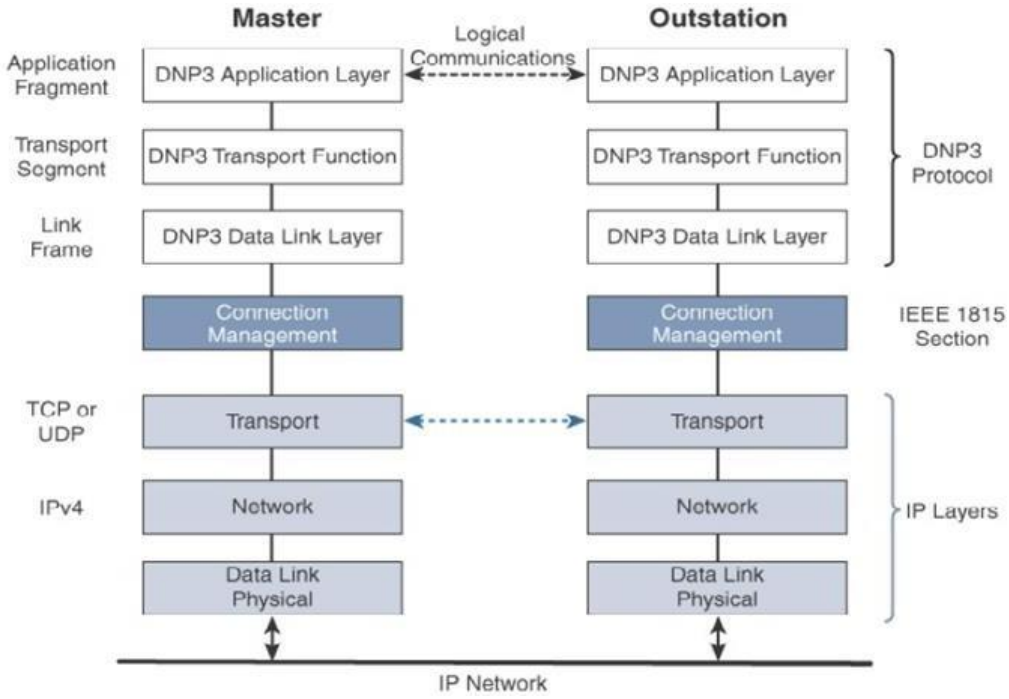


Figure 2.18: Protocol Stack for Transporting Serial DNP3 SCADA over IP

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

IoT Application Layer Protocols

When considering constrained networks and/or a large-scale deployment of constrained nodes, verbose web-based and data model protocols, may be too heavy for IoT applications. To address this problem, the IoT industry is working on new lightweight protocols that are better suited to large numbers of constrained nodes and networks. Two of the most popular

protocols are CoAP and MQTT. Figure 2.19 highlights their position in a common IoT protocol stack.

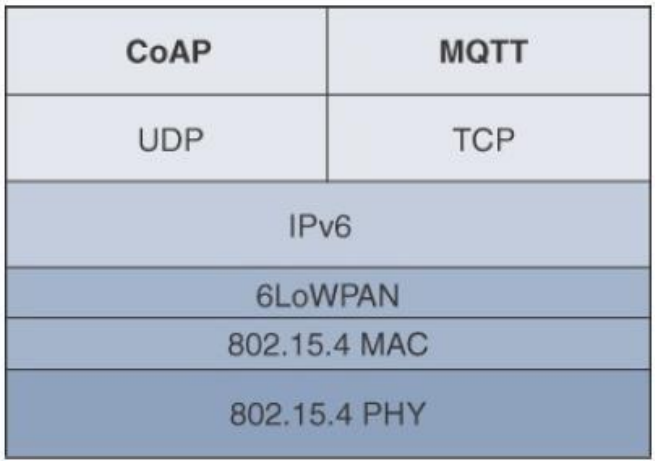


Figure 2.19: Example of a High-Level IoT Protocol Stack for CoAP and MQTT

In Figure 2.19, CoAP and MQTT are naturally at the top of this sample IoT stack, based on an IEEE 802.15.4 mesh network. While there are a few exceptions, you will almost always find CoAP deployed over UDP and MQTT running over TCP. The following sections take a deeper look at CoAP and MQTT.

CoAP

Constrained Application Protocol (CoAP) resulted from the IETF Constrained RESTful Environments (CoRE) working group's efforts to develop a generic framework for resource-oriented applications targeting constrained nodes and networks. The CoAP framework defines simple and flexible ways to manipulate sensors and actuators for data or device management.

The CoAP messaging model is primarily designed to facilitate the exchange of messages over UDP between endpoints, including the secure transport protocol Datagram Transport Layer Security (DTLS).

From a formatting perspective, a CoAP message is composed of a short fixed-length Header field (4 bytes), a variable-length but mandatory Token field (0–8 bytes), Options fields if necessary, and the Payload field. Figure 2.20 details the CoAP message format, which delivers low overhead while decreasing parsing complexity.

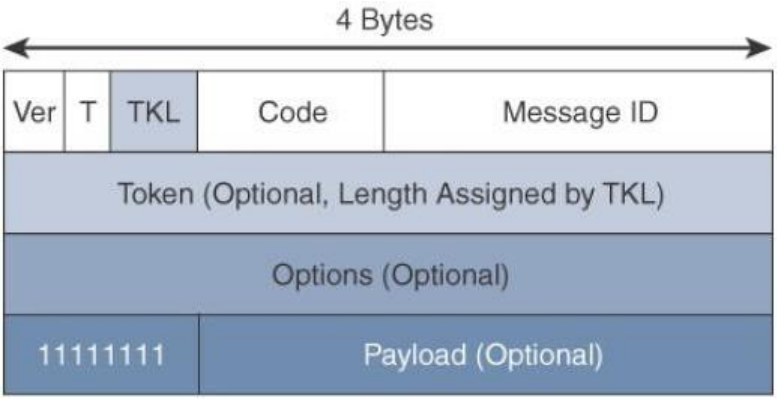


Figure 2.20: CoAP Message Format

The CoAP message format is relatively simple and flexible. It allows CoAP to deliver low overhead, which is critical for constrained networks, while also being easy to parse and process for constrained devices.

Table 2.4 CoAP Message Fields

CoAP Message Field	Description
Ver (Version)	Identifies the CoAP version.
T (Type)	Defines one of the following four message types: Confirmable (CON), Non-confirmable (NON), Acknowledgement (ACK), or Reset (RST). CON and ACK are highlighted in more detail in Figure 6-9.
TKL (Token Length)	Specifies the size (0–8 Bytes) of the Token field.
Code	Indicates the request method for a request message and a response code for a response message. For example, in Figure 6-9, GET is the request method, and 2.05 is the response code. For a complete list of values for this field, refer to RFC 7252.
Message ID	Detects message duplication and used to match ACK and RST message types to Con and NON message types.
Token	With a length specified by TKL, correlates requests and responses.
Options	Specifies option number, length, and option value. Capabilities provided by the Options field include specifying the target resource of a request and proxy functions.
Payload	Carries the CoAP application data. This field is optional, but when it is present, a single byte of all 1s (0xFF) precedes the payload. The purpose of this byte is to delineate the end of the Options field and the beginning of Payload.

CoAP can run over IPv4 or IPv6. However, it is recommended that the message fit within a single IP packet and UDP payload to avoid fragmentation. For IPv6, with the default MTU size being 1280 bytes and allowing for no fragmentation across nodes, the maximum CoAP message size could be up to 1152 bytes, including 1024 bytes for the payload. In the

case of IPv4, as IP fragmentation may exist across the network, implementations should limit themselves to more conservative values and set the IPv4 Don't Fragment (DF) bit.

CoAP communications across an IoT infrastructure can take various paths. Connections can be between devices located on the same or different constrained networks or between devices and generic Internet or cloud servers, all operating over IP. Proxy mechanisms are also defined, and RFC 7252 details a basic HTTP mapping for CoAP. As both HTTP and CoAP are IP-based protocols, the proxy function can be located practically anywhere in the network, not necessarily at the border between constrained and non-constrained networks.

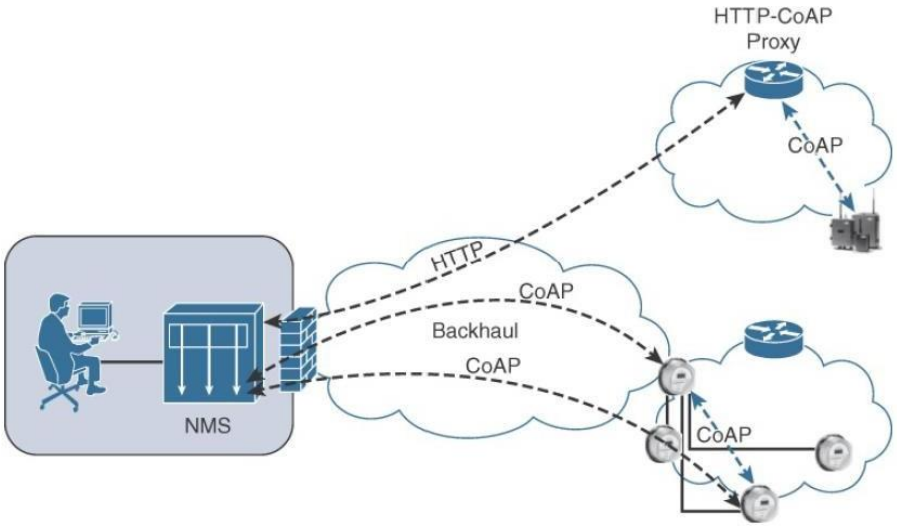


Figure 2.21: CoAP Communications in IoT Infrastructures

Just like HTTP, CoAP is based on the REST architecture, but with a "thing" acting as both the client and the server. Through the exchange of asynchronous messages, a client requests an action via a method code on a server resource. A uniform resource identifier (URI) localized on the server identifies this resource. The server responds with a response code that may include a resource representation. The CoAP request/response semantics include the methods GET, POST, PUT, and DELETE.

Message Queuing Telemetry Transport (MQTT)

At the end of the 1990s, engineers from IBM and Arcom (acquired in 2006 by Eurotech) were looking for a reliable, lightweight, and cost-effective protocol to monitor and control a large number of sensors and their data from a central server location, as typically used by the oil and gas industries. Their research resulted in the development and implementation of the Message Queuing Telemetry Transport (MQTT) protocol that is now standardized by the Organization for the Advancement of Structured Information Standards (OASIS).

The selection of a client/server and publish/subscribe framework based on the TCP/IP architecture, as shown in Figure 2.22

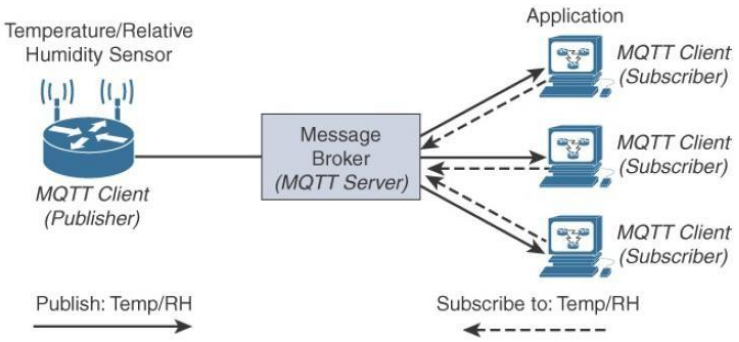


Figure 2.22: MQTT Publish/Subscribe Framework

An MQTT client can act as a publisher to send data (or resource information) to an MQTT server acting as an MQTT message broker. In the example illustrated in Figure 2.22, the MQTT client on the left side is a temperature (Temp) and relative humidity (RH) sensor that publishes its Temp/RH data. The MQTT server (or message broker) accepts the network connection along with application messages, such as Temp/RH data, from the publishers. It also handles the subscription and unsubscription process and pushes the application data to MQTT clients acting as subscribers.

The application on the right side of Figure 2.22 is an MQTT client that is a subscriber to the Temp/RH data being generated by the publisher or sensor on the left. This model, where subscribers express a desire to receive information from publishers, is well known. A great example is the collaboration and social networking application Twitter.

With MQTT, clients can subscribe to all data (using a wildcard character) or specific data from the information tree of a publisher. In addition, the presence of a message broker in MQTT decouples the data transmission between clients acting as publishers and subscribers. In fact, publishers and subscribers do not even know (or need to know) about each other. A benefit of having this decoupling is that the MQTT message broker ensures that information can be buffered and cached in case of network failures. This also means that publishers and subscribers do not have to be online at the same time. MQTT control packets run over a TCP transport using port 1883. TCP ensures an ordered, lossless stream of bytes between the MQTT client and the MQTT server. Optionally, MQTT can be secured using TLS on port 8883, and WebSocket (defined in RFC 6455) can also be used.

MQTT is a lightweight protocol because each control packet consists of a 2-byte fixed header with optional variable header fields and optional payload. You should note that a control packet can contain a payload up to 256 MB. Figure 2.23 provides an overview of the MQTT message format.

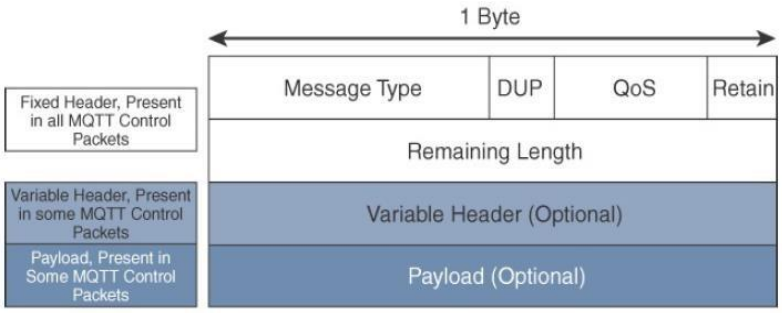


Figure 2.23 *MQTT Message Format*

Compared to the CoAP message format, MQTT contains a smaller header of 2 bytes compared to 4 bytes for CoAP. The first MQTT field in the header is Message Type, which identifies the kind of MQTT packet within a message. Fourteen different types of control packets are specified in MQTT version 3.1.1. Each of them has a unique value that is coded into the Message Type field. Note that values 0 and 15 are reserved. MQTT message types are summarized in Table 2.5.

Table 2.5 MQTT Message Type

Message Type	Value	Flow	Description
CONNECT	1	Client to server	Request to connect
CONNACK	2	Server to client	Connect acknowledgement
PUBLISH	3	Client to server Server to client	Publish message
PUBACK	4	Client to server Server to client	Publish acknowledgement
PUBREC	5	Client to server Server to client	Publish received
PUBREL	6	Client to server Server to client	Publish release
PUBCOMP	7	Client to server Server to client	Publish complete
SUBSCRIBE	8	Client to server	Subscribe request
SUBACK	9	Server to client	Subscribe acknowledgement
UNSUBSCRIBE	10	Client to server	Unsubscribe request
UNSUBACK	11	Server to client	Unsubscribe acknowledgement
PINGREQ	12	Client to server	Ping request
PINGRESP	13	Server to client	Ping response
DISCONNECT	14	Client to server	Client disconnecting

The next field in the MQTT header is DUP (Duplication Flag). This flag, when set, allows the client to notate that the packet has been sent previously, but an acknowledgement was not received. The QoS header field allows for the selection of three different QoS levels. The next field is the Retain flag. Only found in a PUBLISH message, the Retain flag notifies the server to hold onto the message data. This allows new subscribers to instantly receive the last known value without having to wait for the next update from the publisher.

The last mandatory field in the MQTT message header is Remaining Length. This field specifies the number of bytes in the MQTT packet following this field.

MQTT sessions between each client and server consist of four phases: session establishment, authentication, data exchange, and session termination. Each client connecting to a server has a unique client ID, which allows the identification of the MQTT session between both parties. When the server is delivering an application message to more than one client, each client is treated independently.

Subscriptions to resources generate SUBSCRIBE/SUBACK control packets, while unsubscription is performed through the exchange of UNSUBSCRIBE/UNSUBACK control packets. Graceful termination of a connection is done through a DISCONNECT control packet, which also offers the capability for a client to reconnect by re-sending its client ID to resume the operations.

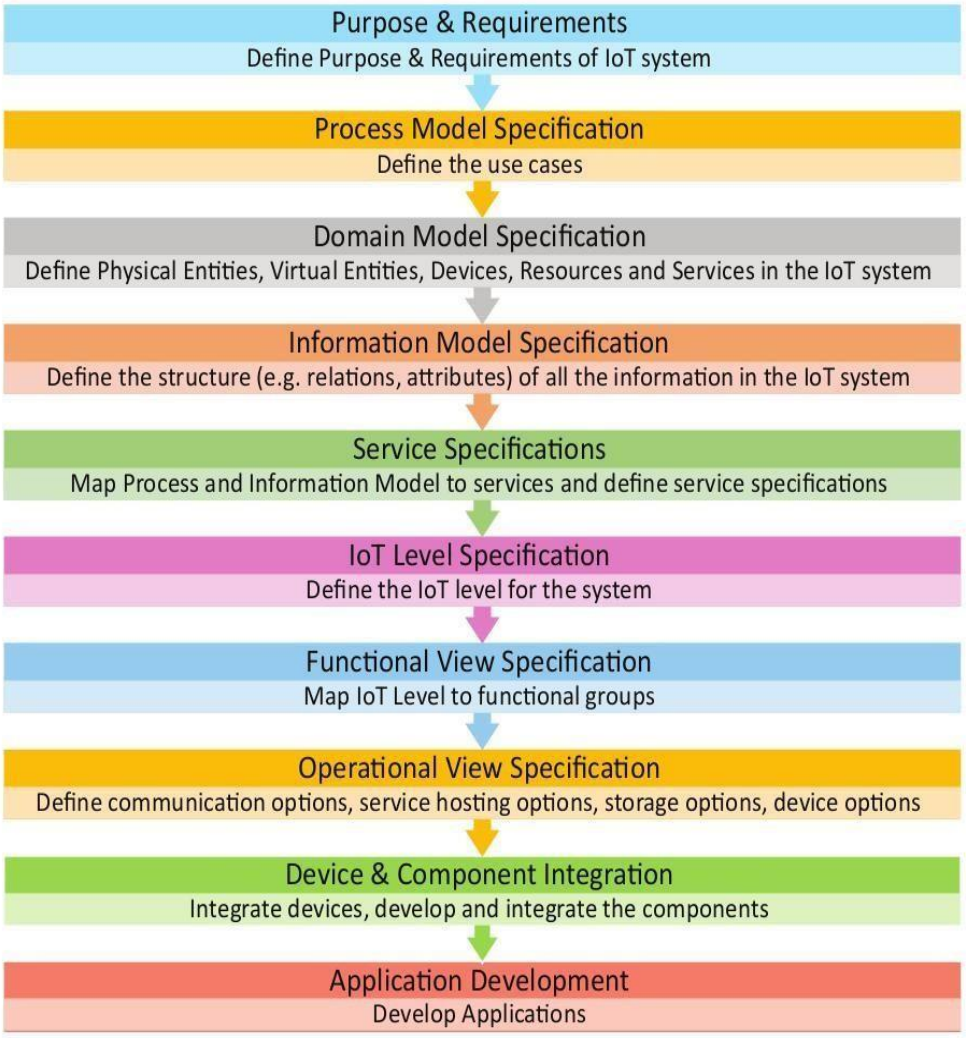
A message broker uses a topic string or topic name to filter messages for its subscribers. When subscribing to a resource, the subscriber indicates the one or more topic levels that are used to structure the topic name. The forward slash (/) in an MQTT topic name is used to separate each level within the topic tree and provide a hierarchical structure to the topic names.

Comparison of CoAP and MQTT

Factor	CoAP	MQTT
Main transport protocol	UDP	TCP
Typical messaging	Request/response	Publish/subscribe
Effectiveness in LLNs	Excellent	Low/fair (Implementations pairing UDP with MQTT are better for LLNs.)
Security	DTLS	SSL/TLS
Communication model	One-to-one	many-to-many
Strengths	Lightweight and fast, with low overhead, and suitable for constrained networks; uses a RESTful model that is easy to code to; easy to parse and process for constrained devices; support for multicasting; asynchronous and synchronous messages	TCP and multiple QoS options provide robust communications; simple management and scalability using a broker architecture
Weaknesses	Not as reliable as TCP-based MQTT, so the application must ensure reliability.	Higher overhead for constrained devices and networks; TCP connections can drain low-power devices; no multicasting support

DESIGN AND DEVELOPMENT

IoT Design Methodology – Steps



Step 1: Purpose & Requirements Specification • The first step in IoT system design methodology is to define the purpose and requirements of the system. In this step, the system purpose, behavior and requirements (such as data collection requirements, data analysis requirements, system management requirements, data privacy and security requirements, user interface requirements, ...) are captured.

Step 2: Process Specification • The second step in the IoT design methodology is to define the process specification. In this step, the use cases of the IoT system are formally described based on and derived from the purpose and requirement specifications.

Step 3: Domain Model Specification • The third step in the IoT design methodology is to define the Domain Model. The domain model describes the main concepts, entities and objects in the domain of IoT system to be designed. Domain model defines the attributes of the objects and relationships between objects. Domain model provides an abstract representation of the concepts, objects and entities in the IoT domain, independent of any specific technology or platform. With the domain model, the IoT system designers can get an understanding of the IoT domain for which the system is to be designed.

Step 4: Information Model Specification • The fourth step in the IoT design methodology is to define the Information Model. Information Model defines the structure of all the information in the IoT system, for example, attributes of Virtual Entities, relations, etc. Information model does not describe the specifics of how the information is represented or stored. To define the information model, we first list the Virtual Entities defined in the Domain Model. Information model adds more details to the Virtual Entities by defining their attributes and relations.

Step 5: Service Specifications • The fifth step in the IoT design methodology is to define the service specifications. Service specifications define the services in the IoT system, service types, service inputs/output, service endpoints, service schedules, service preconditions and service effects.

Step 6: IoT Level Specification • The sixth step in the IoT design methodology is to define the IoT level for the system.

Step 7: Functional View Specification • The seventh step in the IoT design methodology is to define the Functional View. The Functional View (FV) defines the functions of the IoT systems grouped into various Functional Groups (FGs). Each Functional Group either provides functionalities for interacting with instances of concepts defined in the Domain Model or provides information related to these concepts.

Step 8: Operational View Specification • The eighth step in the IoT design methodology is to define the Operational View Specifications. In this step, various options pertaining to the IoT system deployment and operation are defined, such as, service hosting options, storage options, device options, application hosting options, etc

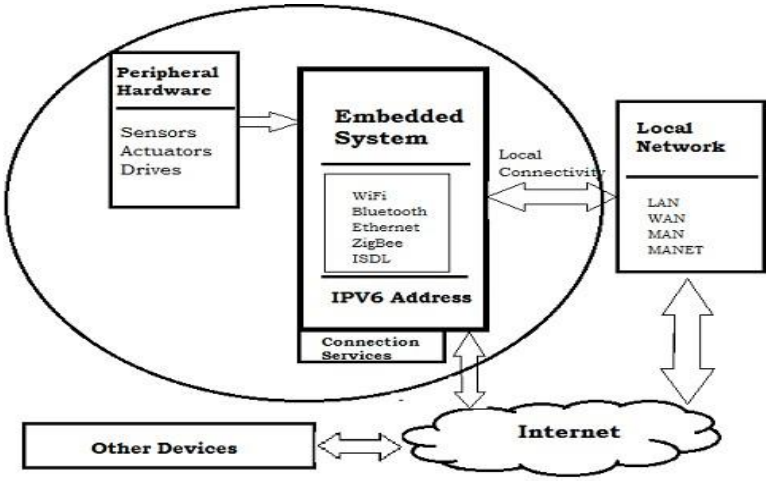
Step 9: Device & Component Integration • The ninth step in the IoT design methodology is the integration of the devices and components.

Step 10: Application Development • The final step in the IoT design methodology is to develop the IoT application.

Embedded Computing Logic

It is essential to know about the embedded devices while learning the IoT or building the projects on IoT. The embedded devices are the objects that build the unique computing system. These systems may or may not connect to the Internet.

An embedded device system generally runs as a single application. However, these devices can connect through the internet connection, and able communicate through other network devices.



Internet of Things (IoT)

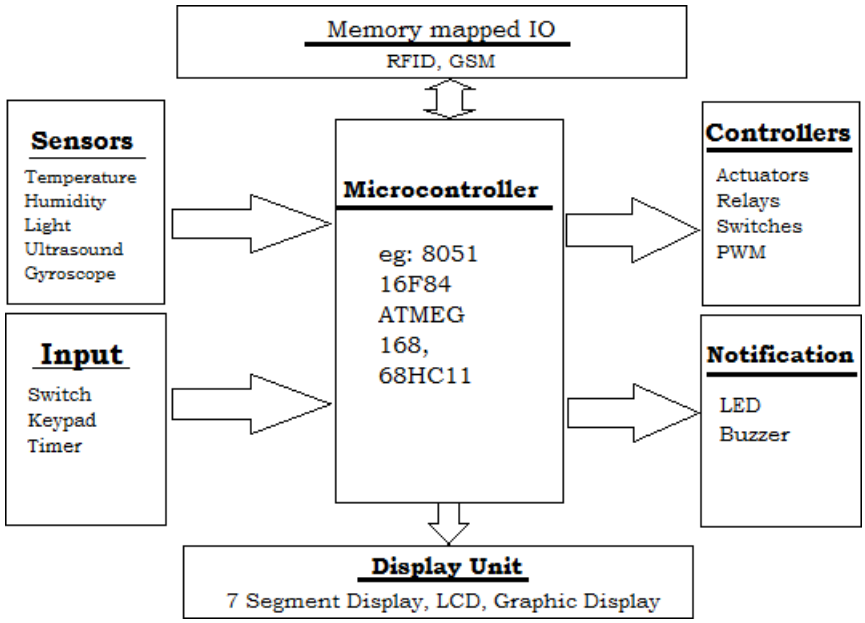
First developed in the 1960s for aerospace and the military, embedded computing systems continue to support new applications through numerous feature enhancements and cost-to-performance improvements of microcontrollers and programmable logic devices. Today, embedded computing systems control everyday devices which we don't generally think of as "computers": digital cameras, automobiles, smart watches, home appliances, and even smart garments. These embedded computing systems are commonly found in consumer, industrial, automotive, medical, commercial, and military applications.

Unlike general-purpose computers, embedded control systems are typically designed to perform specific tasks. The embedded computing system designer's task is to identify the set of components that will implement the system's functional, performance, usability, and reliability requirements, typically within tight cost and development timeline constraints. Accordingly, the selection of a microcontroller and its characteristics, including data processing capabilities, speed, peripherals, and power consumption, is one of the earliest and most critical aspects of system design.

Part of the designer's responsibility involves being aware of trends in their particular industry and taking advantage of relevant components and techniques . Let's look for examples among the top industries for microcontroller applications, the Internet of Things.

Embedded System Hardware

The embedded system can be of type microcontroller or type microprocessor. Both of these types contain an integrated circuit (IC). The essential component of the embedded system is a RISC family microcontroller like Motorola 68HC11, PIC 16F84, Atmel 8051 and many more. The most important factor that differentiates these microcontrollers with the microprocessor like 8085 is their internal read and writable memory. The essential embedded device components and system architecture are specified below.



Embedded System Software

The embedded system that uses the devices for the operating system is based on the language platform, mainly where the real-time operation would be performed. Manufacturers build embedded software in electronics, e.g., cars, telephones, modems, appliances, etc. The embedded system software can be as simple as lighting controls running using an 8-bit microcontroller. It can also be complicated software for missiles, process control systems, airplanes etc.

Microcontrollers for Embedded Computing with IoT Devices

IoT devices are meant to be inexpensive, therefore the microcontroller needs to be chosen so that its capabilities are not underutilized by the application. The microcontroller specifications that determine the best part for your application are:

- **Bit depth:** The register and data path width impacts the speed and accuracy with which microcontrollers can perform non-trivial computations.
- **Memory:** The amount of RAM and Flash in a microcontroller determines the code size and complexity the component can support at full speed. Large memories have larger die area and component cost.

- **GPIO:** These are the microcontroller pins used to connect to sensors and actuators in the system. These often share their functionality with other microcontroller peripherals, such as serial communication, A/D, and D/A converters.
- **Power consumption:** Power consumption is critically important for battery-operated devices and it typically increases with microcontroller speed and memory size.

System on Chips

System on Chip in IoT designed by Redpine Signals is discussed below.This IoT SoC supports WLAN, bluetooth and Zigbee systems on a single chip. It also supports 2.4 and 5GHz radio frequencies.

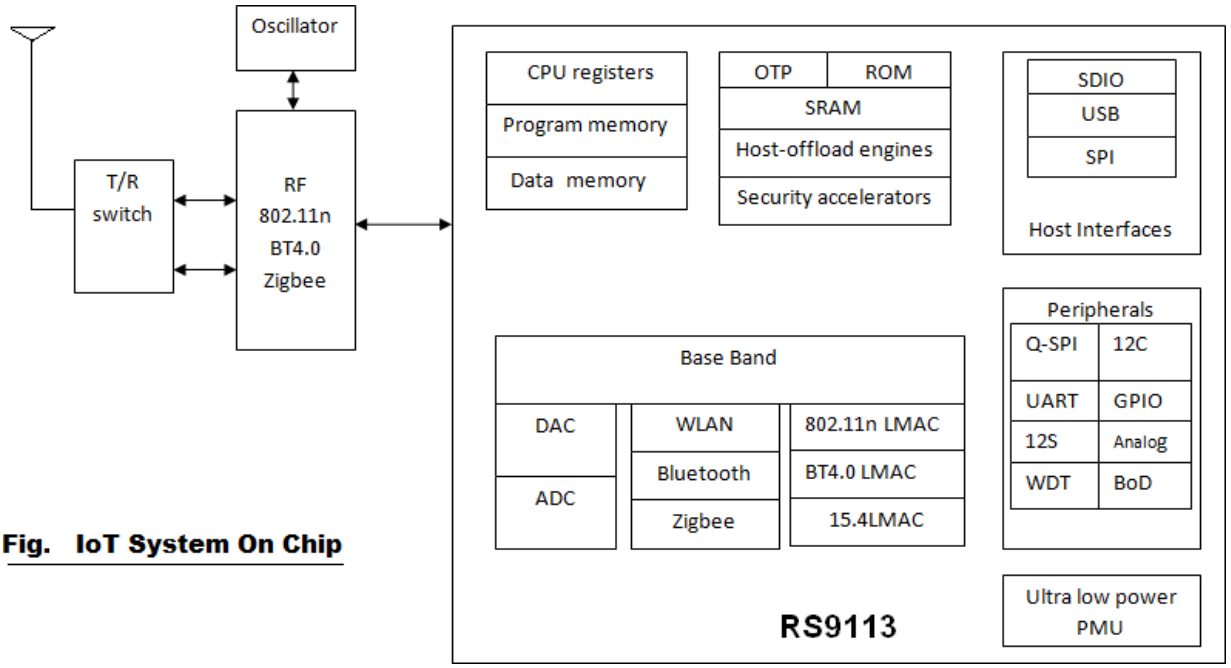


Fig. IoT System On Chip

As we know IoT is the technology which will provide communication between things, between things and people using internet and IP enabled protocols. As we have seen in IoT tutorial any IoT compliant system will have two major parts viz. front end and back end. Front end provides connectivity with physical world and consists of sensors while backend consists of processing and network connectivity interfaces.

Typical **IoT system on chip** support more than one RATs (Radio Access Technologies). It will have following modules.

- Transmit and receive switch.

• RF part mainly consists of Trasmitter, receiver, oscillator and amplifiers.
• Memoriesi.e. Program memory, data memory to store the code and data
• Physical layer(baseband processing) either on FPGA or on processor based on complexity and latency requirement.
• MAC layer and upper protocol stacks TCP/IP etc. running on processor
• ADC and DAC to provide interface between digital baseband and analog RF portions.
• Various interfaces such as SDIO, USB, SPI etc to provide interface with the host.
• Other peripherals such as UART, I^2C, GPIO, WDT etc. to use the IoT SoC for various connections.

As IoT system on chip supports multiple wireless protocols and RF hardware to support multiple frequency bands, following factors need to be carefully analyzed and to be optimized.

- Power-consumption
- Data-throughput
- Device-size
- Performance in terms of latency and other factors

Figure depicts one such IoT System on Chip model no. RS9113,which has been designed and developed by Redpine Signals recently. It supports WLAN (802.11n), Bluetooth version 4.0 and Zigbee (802.15.4-2006) in the same chip. Hence the IoT device can be connected with any of the said wireless technology based networks.

This IoT SoC (system on chip in IoT) can be used for numerous applications as mentioned below:

- Mobile
- M2M-Communication
- Real time location finding tags
- Thermostats
- Smart meters
- Wireless sensor devices
- Serial to WiFi converter
- Voice Over WiFi compliant phones
- Home automation
- Health care devices and equipments

Building Blocks Of IoT

Four things form basic building blocks of the IoT system –sensors, processors, gateways, applications. Each of these nodes has to have its own characteristics in order to form an useful IoT system.

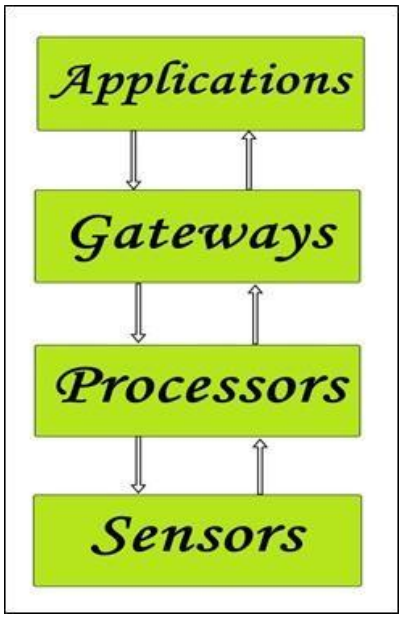


Figure: Simplified block diagram of the basic building blocks of the **IoT**

Sensors:

- These form the front end of the IoT devices. These are the so-called "Things" of the system. Their main purpose is to collect data from its surroundings (sensors) or give out data to its surrounding (actuators).
- These have to be uniquely identifiable devices with a unique IP address so that they can be easily identifiable over a large network.
- These have to be active in nature which means that they should be able to collect real-time data. These can either work on their own (autonomous in nature) or can be made to work by the user depending on their needs (user-controlled).
- Examples of sensors are gas sensor, water quality sensor, moisture sensor, etc.

Processors:

- Processors are the brain of the IoT system. Their main function is to process the data captured by the sensors and process them so as to extract the valuable data from the enormous amount of raw data collected. In a word, we can say that it gives intelligence to the data.
- Processors mostly work on real-time basis and can be easily controlled by applications. These are also responsible for securing the data – that is performing encryption and decryption of data.
- Embedded hardware devices, microcontroller, etc are the ones that process the data because they have processors attached to it.

Gateways:

- Gateways are responsible for routing the processed data and send it to proper locations for its (data) proper utilization.
- In other words, we can say that gateway helps in to and fro communication of the data. It provides network connectivity to the data. Network connectivity is essential for any IoT system to communicate.
- LAN, WAN, PAN, etc are examples of network gateways.

Applications:

- Applications form another end of an IoT system. Applications are essential for proper utilization of all the data collected.
- These cloud-based applications which are responsible for rendering the effective meaning to the data collected. Applications are controlled by users and are a delivery point of particular services.
- Examples of applications are home automation apps, security systems, industrial control hub, etc.

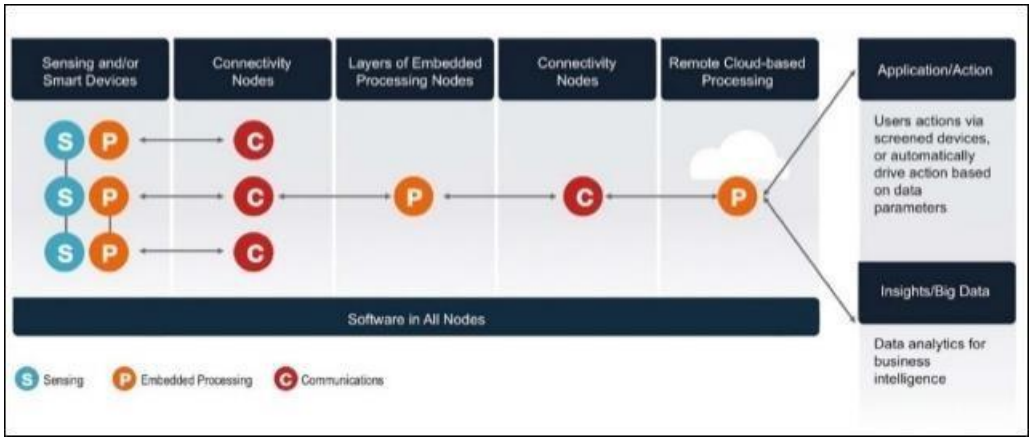


Figure : Basic building blocks of IoT

In a nutshell, from the figure we can determine that the information gathered by the sensing node (end node) is processed first then via connectivity it reaches the embedded processing nodes that can be any embedded hardware devices and are processed there as well. It then passes through the connectivity nodes again and reaches the remote cloud-based processing that can be any software and is sent to the application node for the proper applied usage of the data collected and also for data analysis via big data.

HOW IoT WORKS

How the IoT works is quite simple.

First, it acquires information with respect to basic resources (names, addresses and so on) and related attributes of objects by means of automatic identification and perception technologies such as RFID, wireless sensor and satellite positioning, in other words, the

sensors, RFID tags, and all other uniquely identifiable objects or "things" acquire real-time information (data) with the virtue of a central hub like smartphones.

Second, by virtue of many kinds of communications technologies, it integrates object-related information into the information network and realizes the intelligent indexing and integration of the information related to masses of objects by resorting to fundamental resource services (similar to the resolution, addressing and discovery of the internet).

Finally, utilizing intelligent computing technologies such as cloud computing, fuzzy recognition, data mining, and semantic analysis, it analyzes and processes the information related to masses of objects so as to eventually realize intelligent decision and control in the physical world.

Let’s have a look at the following diagram.

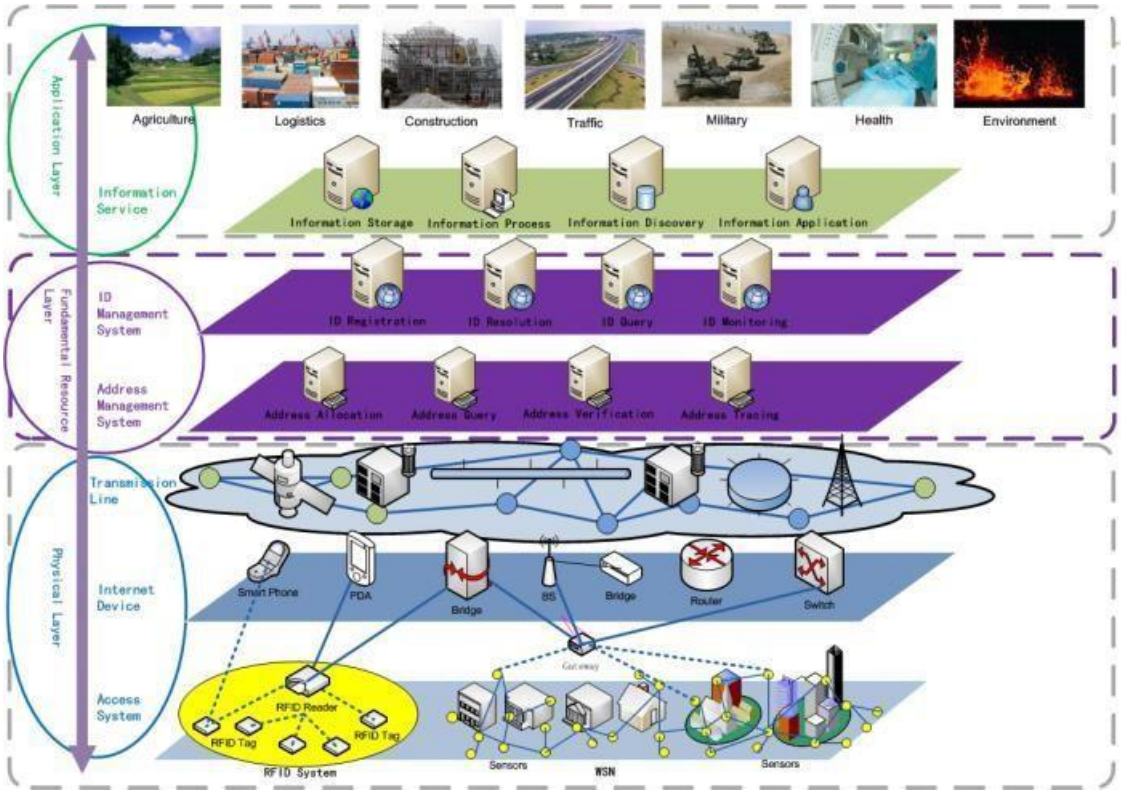
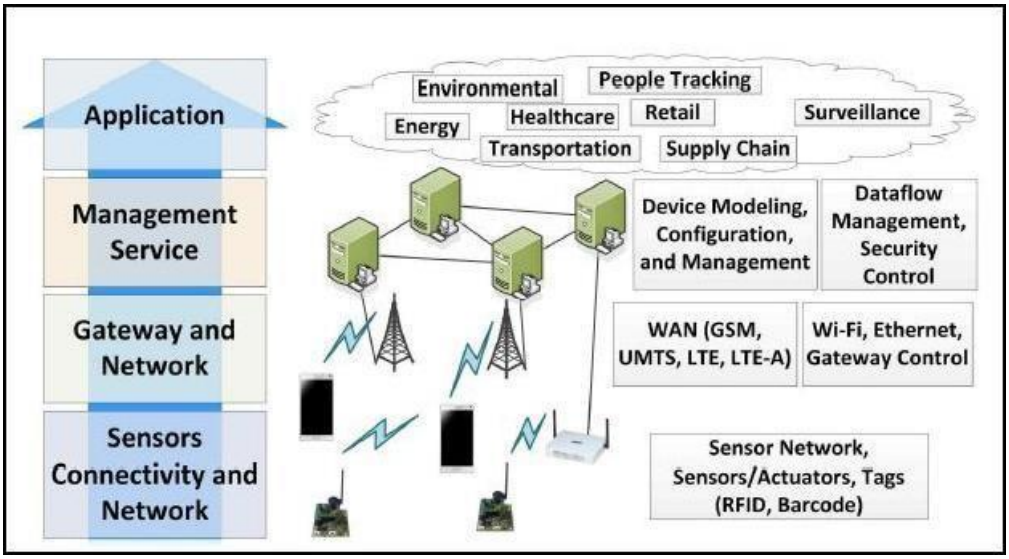


Figure : Layers of the IoT

In the Physical layer, all the data collected by the access system (uniquely identifiable "things") collect data and go to the internet devices (like smartphones). Then via transmission lines (like fiber-optic cable) it goes to the management layer where all the data is managed separately (stream analytics and data analytics) from the raw data. Then all the managed information is released to the application layer for proper utilization of the data collected.

IoT Architecture Layers

There are four major layers.



At the very bottom of IoT architecture, we start with the Sensors and Connectivity network which collects information. Then we have the Gateway and Network Layer. Above which we have the Management Service layer and then at the end, we have the application layer where the data collected are processed according to the needs of various applications.

Let's discuss the features of each of these architectural layers separately.

Sensor, Connectivity and Network Layer

- This layer consists of RFID tags, sensors (which are an essential part of an IoT system and are responsible for collecting raw data). These form the essential "things" of an IoT system.
- Sensors, RFID tags are wireless devices and form the Wireless Sensor Networks (WSN).
- Sensors are active in nature which means that real-time information is to be collected and processed.
- This layer also has the network connectivity (like WAN, PAN, etc.) which is responsible for communicating the raw data to the next layer which is the Gateway and Network Layer.
- The devices which are comprised of WSN have finite storage capacity, restricted communication bandwidth and have small processing speed.
- We have different sensors for different applications – temperature sensor for collecting temperature data, water quality for examining water quality, moisture sensor for measuring moisture content of the atmosphere or soil, etc.

As per the figure below, at the bottom of this layer, we have the tags which are the RFID tags or barcode reader, above which we have the sensors/actuators and then the communication networks.

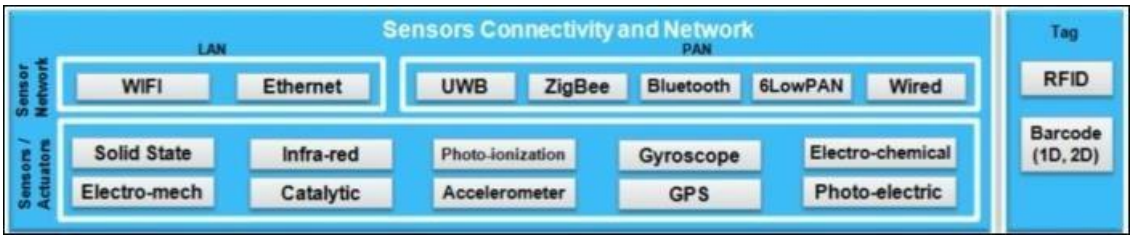


Figure : Sensor, Connectivity and Network Layer

Gateway and Network Layer

- Gateways are responsible for routing the data coming from the **Sensor, Connectivity and Network layer** and pass it to the next layer which is the **Management Service Layer**.
- This layer requires having a large storage capacity for storing the enormous amount of data collected by the sensors, RFID tags, etc. Also, this layer needs to have a consistently trusted performance in terms of public, private and hybrid networks.
- Different IoT device works on different kinds of network protocols. All these protocols are required to be assimilated into a single layer. This layer is responsible for integrating various network protocols.

From the figure below, at the bottom, we have the gateway which is comprised of the embedded OS, Signal Processors, and Modulators, Micro-Controllers etc. Above the gateway we have the Gateway Networks which are LAN(Local Area Network), WAN(Wide Area Network), etc.

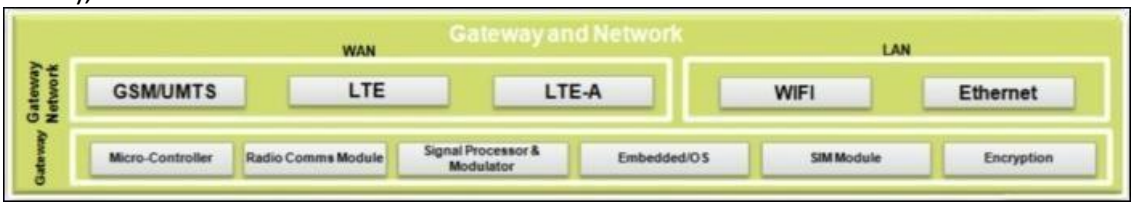


Figure : Gateway and Network Layer

Management Service Layer

- This layer is used for managing IoT services. The management Service layer is responsible for Securing Analysis of IoT devices, Analysis of Information (Stream Analytics, Data Analytics), Device Management.
- Data management is required to extract the necessary information from the enormous amount of raw data collected by the sensor devices to yield a valuable result of all the data collected. This action is performed in this layer.
- Also, a certain situation requires an immediate response to the situation. This layer helps in doing that by abstracting data, extracting information and managing the data flow.
- This layer is also responsible for data mining, text mining, service analytics, etc.

From the figure below, we can see that, management service layer has Operational Support Service (OSS) which includes Device Modeling, Device Configuration and Management and many more. Also, we have the Billing Support System (BSS) which supports billing and reporting.

Also, from the figure, we can see that there are IoT/M2M Application Services which includes Analytics Platform; Data – which is the most important part; Security which includes Access Controls, Encryption, Identity Access Management, etc. ; and then we have the Business Rule Management (BRM) and Business Process Management (BPM).

Figure : Management Service Layer

Application Layer

- Application layer forms the topmost layer of IoT architecture which is responsible for effective utilization of the data collected.
- Various IoT applications include Home Automation, E-health, E-Government, etc.
- From the figure below, we can see that there are two types of applications which are Horizontal Market which includes Fleet Management, Supply Chain, etc. and on the Sector-wise application of IoT we have energy, healthcare, transportation, etc.

Figure : Application Layer

Iot Platform

An IoT platform is a multi-layer technology that enables straightforward provisioning, management, and automation of connected devices within the Internet of Things universe. It basically connects your hardware, however diverse, to the cloud by using flexible connectivity options, enterprise-grade security mechanisms, and broad data processing powers. For developers, an IoT platform provides a set of ready-to-use features that greatly speed up development of applications for connected devices as well as take care of scalability and cross-device compatibility.

Thus, an IoT platform can be wearing different hats depending on how you look at it. It is commonly referred to as middleware when we talk about how it connects remote devices

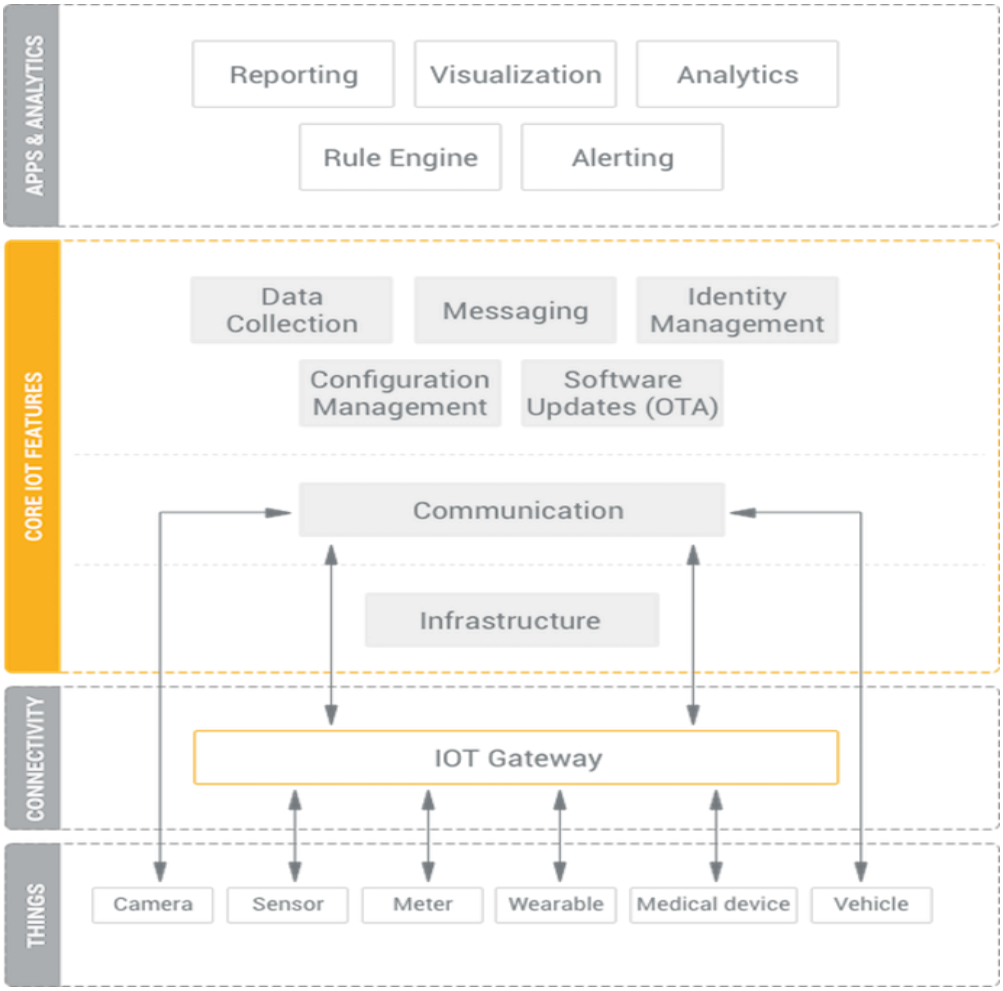
to user applications (or other devices) and manages all the interactions between the hardware and the application layers. It is also known as a cloud enablement platform or IoT enablement platform to pinpoint its major business value, that is empowering standard devices with cloud-based applications and services. Finally, under the name of the IoT application enablement platform, it shifts the focus to being a key tool for IoT developers.

IoT platform as the middleware

IoT platforms originated in the form of IoT middleware, which purpose was to function as a mediator between the hardware and application layers. Its primary tasks included data collection from the devices over different protocols and network topologies, remote device configuration and control, device management, and over-the-air firmware updates.

To be used in real-life heterogeneous IoT ecosystems, IoT middleware is expected to support integration with almost any connected device and blend in with third-party applications used by the device. This independence from underlying hardware and overhanging software allows a single IoT platform to manage any kind of connected device in the same straightforward way.

Modern IoT platforms go further and introduce a variety of valuable features into the hardware and application layers as well. They provide components for frontend and analytics, on-device data processing, and cloud-based deployment. Some of them can handle end-to-end IoT solution implementation from the ground up.



IoT platform technology stack

In the four typical layers of the IoT stack, which are things, connectivity, core IoT features, and applications & analytics, a top-of-the-range IoT platform should provide you with the majority of IoT functionality needed for developing your connected devices and smart things.

Your devices connect to the platform, which sits in the cloud or in your on-premises data center, either directly or by using an IoT gateway. A gateway comes useful whenever your endpoints aren't capable of direct cloud communication or, for example, you need some computing power on edge. You can also use an IoT gateway to convert protocols, for example, when your endpoints are in LoRaWan network but you need them to communicate with the cloud over MQTT.

An IoT platform itself can be decomposed into several layers. At the bottom there is the infrastructure level, which is something that enables the functioning of the platform. You

can find here components for container management, internal platform messaging, orchestration of IoT solution clusters, and others.

The communication layer enables messaging for the devices; in other words, this is where devices connect to the cloud to perform different operations.

The following layer represents core IoT features provided by the platform. Among the essential ones are data collection, device management, configuration management, messaging, and OTA software updates.

Sitting on top of core IoT features, there is another layer, which is less related to data exchange between devices but rather to processing of this data in the platform. There is reporting, which allows you to generate custom reports. There is visualization for data representation in user applications. Then, there are a rule engine, analytics, and alerting for notifying you about any anomalies detected in your IoT solution.

Importantly, the best IoT platforms allow you to add your own industry-specific components and third-party applications. Without such flexibility adapting an IoT platform for a particular business scenario could bear significant extra cost and delay the solution delivery indefinitely.

Advanced IoT platforms

There are some other important criteria that differentiate IoT platforms between each other, such as scalability, customizability, ease of use, code control, integration with 3rd party software, deployment options, and the data security level.

- **Scalable (cloud native)** – advanced IoT platforms ensure elastic scalability across any number of endpoints that the client may require. This capability is taken for granted for public cloud deployments but it should be specifically put to the test in case of an on-premises deployment, including the platform's load balancing capabilities for maximized performance of the server cluster.
- **Customizable** – a crucial factor for the speed of delivery. It closely relates to flexibility of integration APIs, louse coupling of the platform's components, and source code transparency. For small-scale, undemanding IoT solutions good APIs may be enough to fly, while feature-rich, rapidly evolving IoT ecosystems usually require developers to have a greater degree of control over the entire system, its source code, integration interfaces, deployment options, data schemas, connectivity and security mechanisms, etc.
- **Secure** – data security involves encryption, comprehensive identity management, and flexible deployment. End-to-end data flow encryption, including data at rest, device authentication, user access rights management, and private cloud infrastructure for sensitive data – this is the basics of how to avoid potentially compromising breaches in your IoT solution.

Cutting across these aspects, there are two different paradigms of IoT solution cluster deployment offered by IoT platform providers: a public cloud IoT PaaS and a self-hosted private IoT cloud.

IoT cloud enablement

An IoT cloud is a pinnacle of the IoT platforms evolution. Sometimes these two terms are used interchangeably, in which case the system at hand is typically an IoT platform-as-a-service (PaaS). This type of solution allows you to rent cloud infrastructure and an IoT platform all from a single technology provider. Also, there might be ready-to-use IoT solutions (IoT cloud services) offered by the provider, built and hosted on its infrastructure. However, one important capability of a modern IoT platform consists in a private IoT cloud enablement. As opposed to public PaaS solutions located at a provider's cloud, a private IoT cloud can be hosted on any cloud infrastructure, including a private data center. This type of deployment offers much greater control over the new features development, customization, and third-party integrations. It is also advocated for stringent data security and performance requirements.

Arduino is a prototype platform (open-source) based on an easy-to-use hardware and software. It consists of a circuit board, which can be programed (referred to as a microcontroller) and a ready-made software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board.

The key features are −

- Arduino boards are able to read analog or digital input signals from different sensors and turn it into an output such as activating a motor, turning LED on/off, connect to the cloud and many other actions.
- You can control your board functions by sending a set of instructions to the microcontroller on the board via Arduino IDE (referred to as uploading software).
- Unlike most previous programmable circuit boards, Arduino does not need an extra piece of hardware (called a programmer) in order to load a new code onto the board. You can simply use a USB cable.
- Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program.
- Finally, Arduino provides a standard form factor that breaks the functions of the micro-controller into a more accessible package.

Board Types

Various kinds of Arduino boards are available depending on different microcontrollers used. However, all Arduino boards have one thing in common: they are programed through the Arduino IDE.

The differences are based on the number of inputs and outputs (the number of sensors, LEDs, and buttons you can use on a single board), speed, operating voltage, form factor etc. Some boards are designed to be embedded and have no programming interface

(hardware), which you would need to buy separately. Some can run directly from a 3.7V battery, others need at least 5V.

Here is a list of different Arduino boards available.

Arduino boards based on ATMEGA328 microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Uno R3	5V	16MHz	14	6	6	1	USB via ATMega16U2
Arduino Uno R3 SMD	5V	16MHz	14	6	6	1	USB via ATMega16U2
Red Board	5V	16MHz	14	6	6	1	USB via FTDI
Arduino Pro 3.3v/8 MHz	3.3V	8MHz	14	6	6	1	FTDI-Compatible Header
Arduino Pro 5V/16MHz	5V	16MHz	14	6	6	1	FTDI-Compatible Header
Arduino mini 05	5V	16MHz	14	8	6	1	FTDI-Compatible Header
Arduino Pro mini 3.3v/8mhz	3.3V	8MHz	14	8	6	1	FTDI-Compatible Header
Arduino Pro mini 5v/16mhz	5V	16MHz	14	8	6	1	FTDI-Compatible

							Header
Arduino Ethernet	5V	16MHz	14	6	6	1	FTDI-Compatible Header
Arduino Fio	3.3V	8MHz	14	8	6	1	FTDI-Compatible Header
LilyPad Arduino 328 main board	3.3V	8MHz	14	6	6	1	FTDI-Compatible Header
LilyPad Arduino simple board	3.3V	8MHz	9	4	5	0	FTDI-Compatible Header

Arduino boards based on ATMEGA32u4 microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Leonardo	5V	16MHz	20	12	7	1	Native USB
Pro micro 5V/16MHz	5V	16MHz	14	6	6	1	Native USB
Pro micro 3.3V/8MHz	5V	16MHz	14	6	6	1	Native USB
LilyPad Arduino USB	3.3V	8MHz	14	6	6	1	Native USB

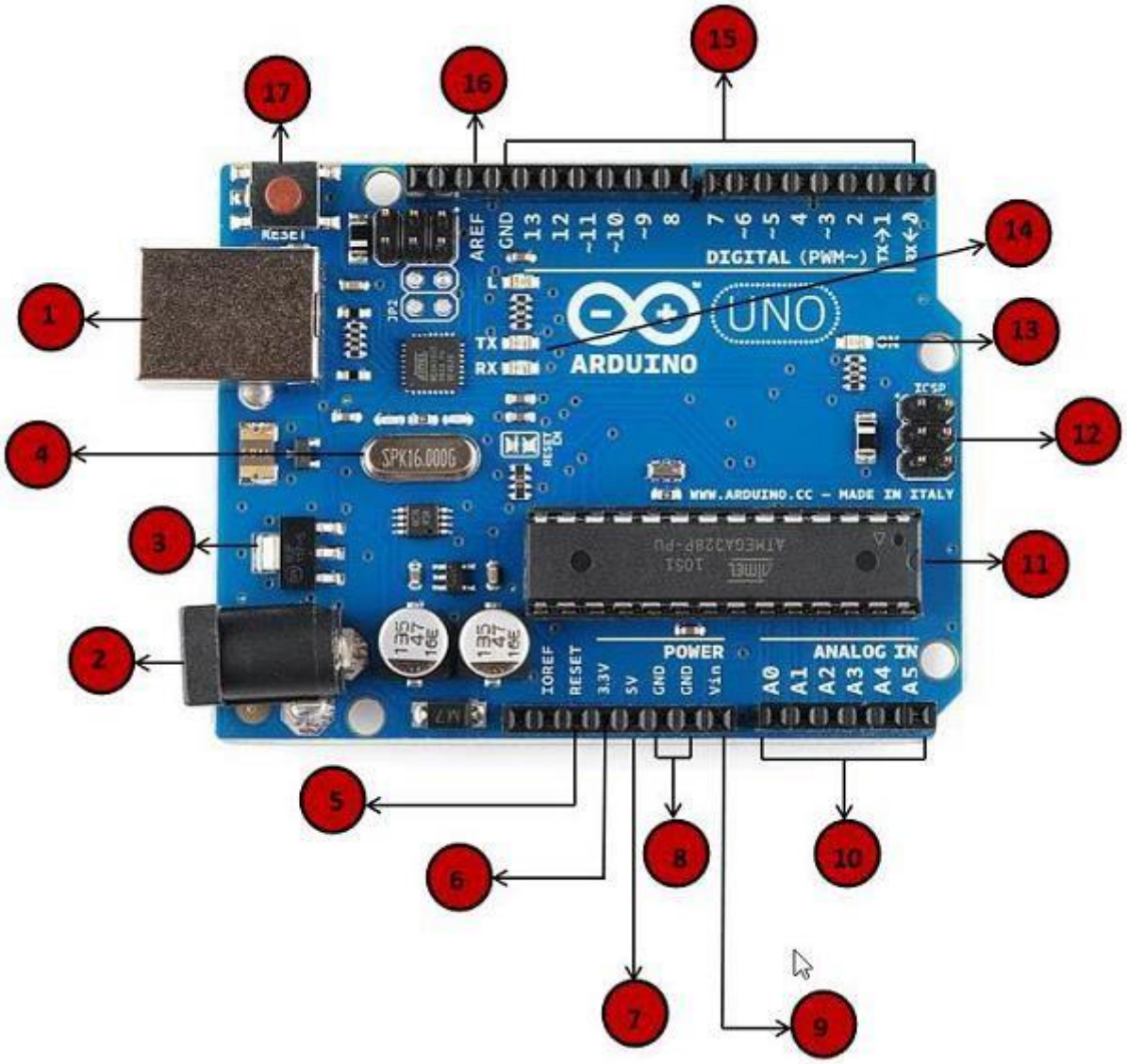
Arduino boards based on ATMEGA2560 microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Mega 2560 R3	5V	16MHz	54	16	14	4	USB via ATMega16U2
Mega Pro 3.3V	3.3V	8MHz	54	16	14	4	FTDI-Compatible Header
Mega Pro 5V	5V	16MHz	54	16	14	4	FTDI-Compatible Header
Mega Pro Mini 3.3V	3.3V	8MHz	54	16	14	4	FTDI-Compatible Header

Arduino boards based on AT91SAM3X8E microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Mega 2560 R3	3.3V	84MHz	54	12	12	4	USB native

In this chapter, we will learn about the different components on the Arduino board. We will study the Arduino UNO board because it is the most popular board in the Arduino board family. In addition, it is the best board to get started with electronics and coding. Some boards look a bit different from the one given below, but most Arduinos have majority of these components in common.



1	**Power USB** Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection (1).
2	**Power (Barrel Jack)** Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (2).
3	**Voltage Regulator** The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.

4	**Crystal Oscillator** The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz.
5,17	**Arduino Reset** You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5).
6,7 8,9	**Pins (3.3, 5, GND, Vin)** • 3.3V (6) − Supply 3.3 output volt • 5V (7) − Supply 5 output volt • Most of the components used with Arduino board works fine with 3.3 volt and 5 volt. • GND (8)(Ground) − There are several GND pins on the Arduino, any of which can be used to ground your circuit. • Vin (9) − This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.
10	**Analog pins** The Arduino UNO board has six analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.
11	**Main microcontroller** Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, you can refer to the data sheet.

	**ICSP pin** Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. Actually, you are slaving the output device to the master of the SPI bus.
	**Power LED indicator** This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.
	**TX and RX LEDs** On your board, you will find two labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication. Second, the TX and RX led (13). The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.
	**Digital I/O** The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled "~" can be used to generate PWM.
	**AREF** AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

After learning about the main parts of the Arduino UNO board, we are ready to learn how to set up the Arduino IDE. Once we learn this, we will be ready to upload our program on the Arduino board.

In this section, we will learn in easy steps, how to set up the Arduino IDE on our computer and prepare the board to receive the program via USB cable.

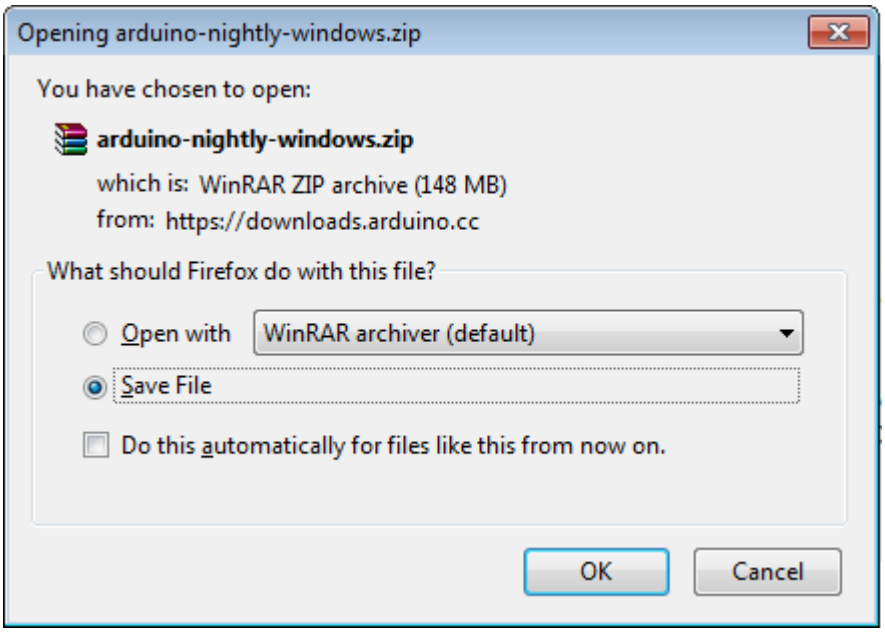
Step 1 − First you must have your Arduino board (you can choose your favorite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega

2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image.

In case you use Arduino Nano, you will need an A to Mini-B cable instead as shown in the following image.

Step 2 – Download Arduino IDE Software.

You can get different versions of Arduino IDE from the Download page on the Arduino Official website. You must select your software, which is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file.



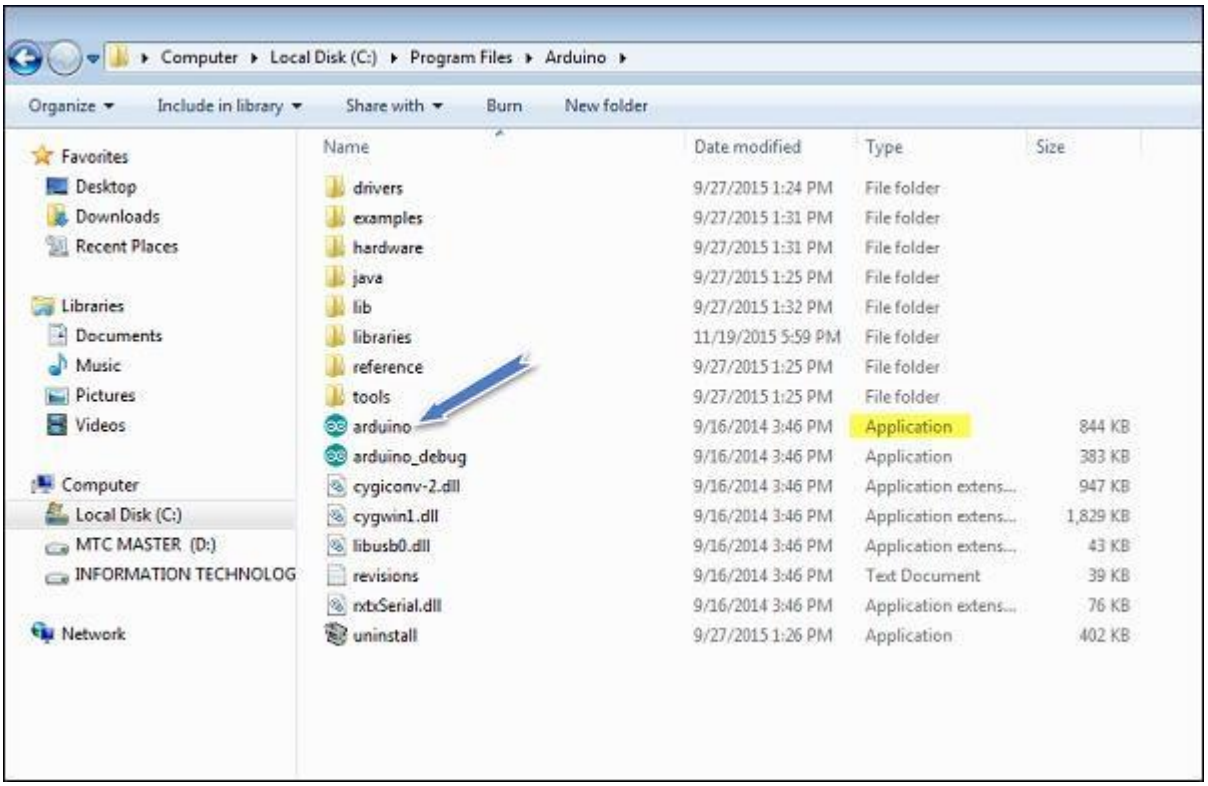
Step 3 – Power up your board.

The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an Arduino Diecimila, you have to make sure that the board is configured to draw power from the USB connection. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port.

Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.

Step 4 – Launch Arduino IDE.

After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.

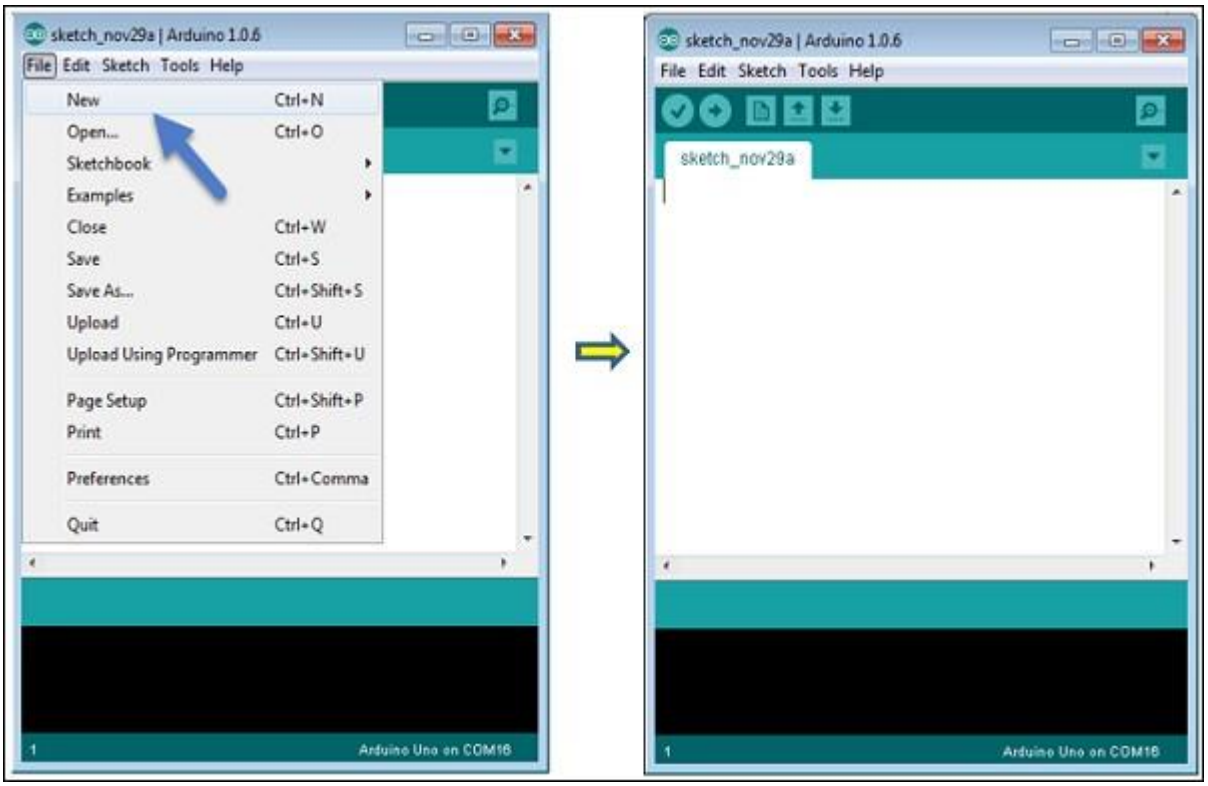


Step 5 – Open your first project.

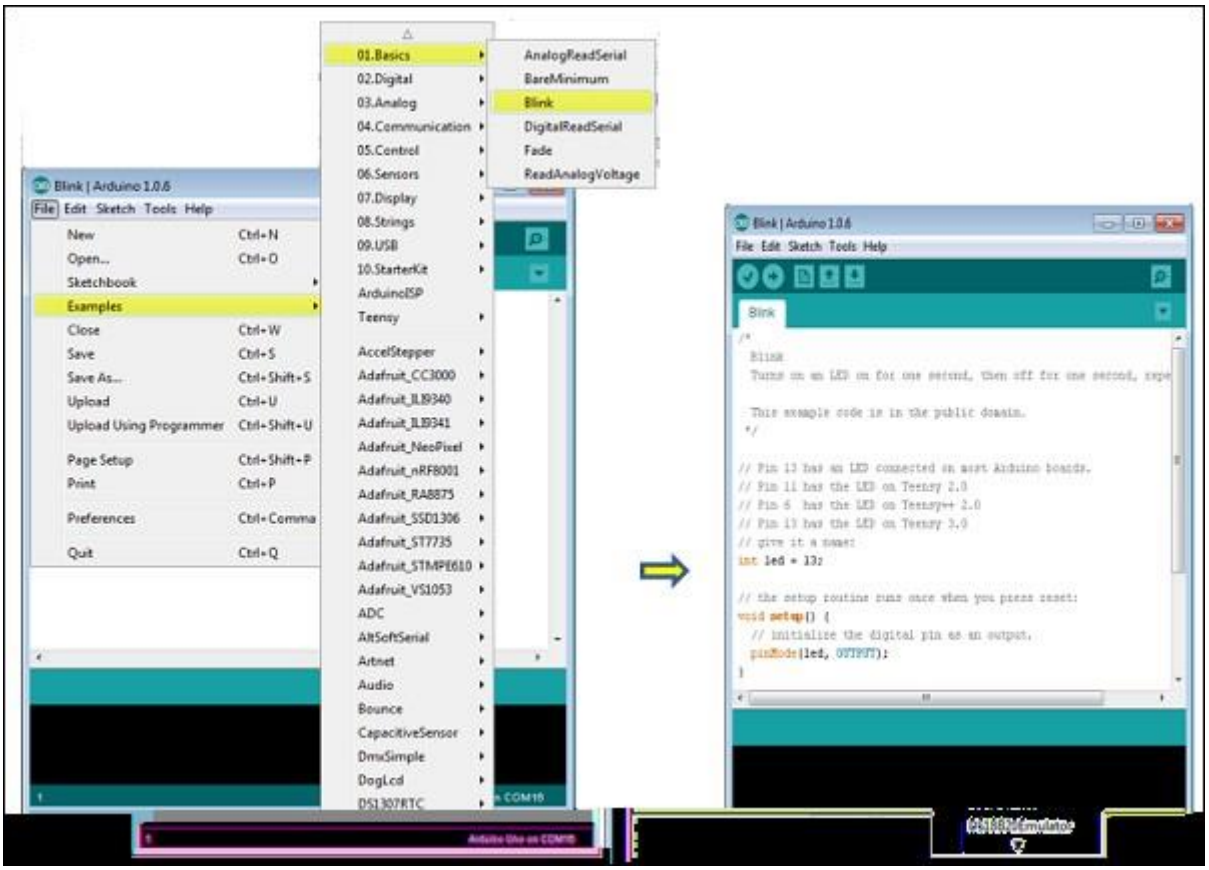
Once the software starts, you have two options −

- Create a new project.
- Open an existing project example.

To create a new project, select File → **New**.



To open an existing project example, select File → Example → Basics → Blink.

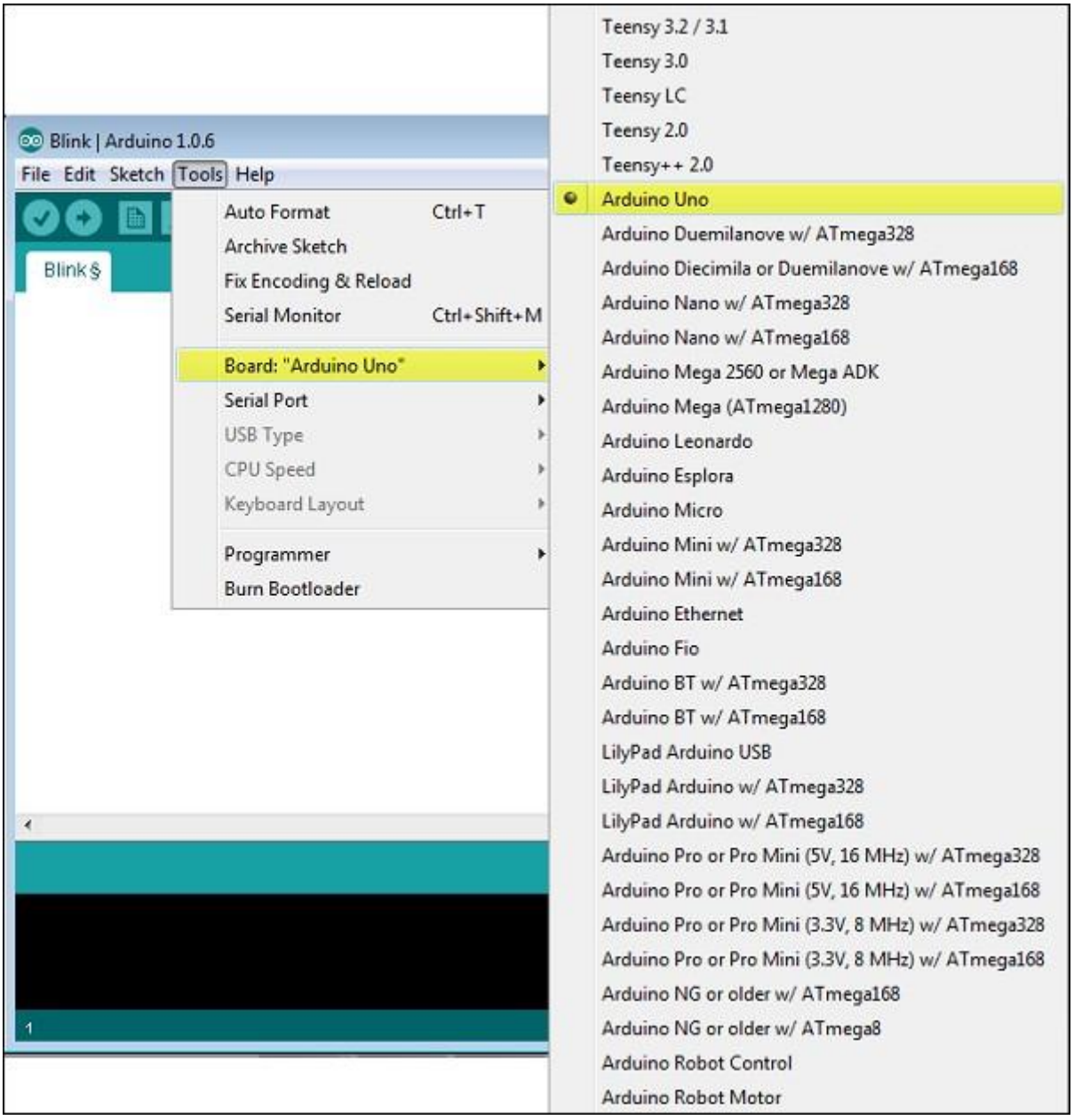


Here, we are selecting just one of the examples with the name **Blink**. It turns the LED on and off with some time delay. You can select any other example from the list.

Step 6 – Select your Arduino board.

To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

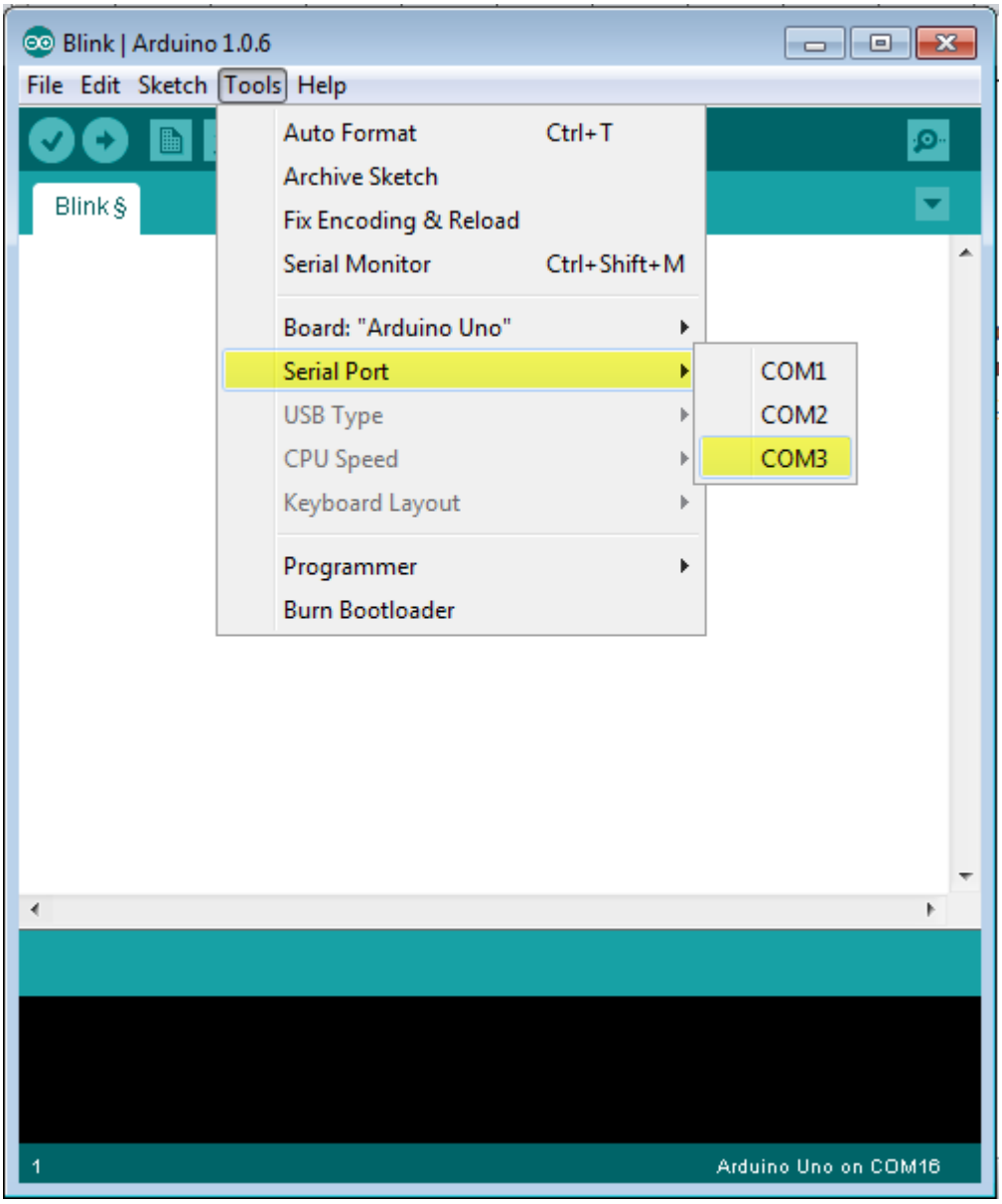
Go to Tools → Board and select your board.



Here, we have selected Arduino Uno board according to our tutorial, but you must select the name matching the board that you are using.

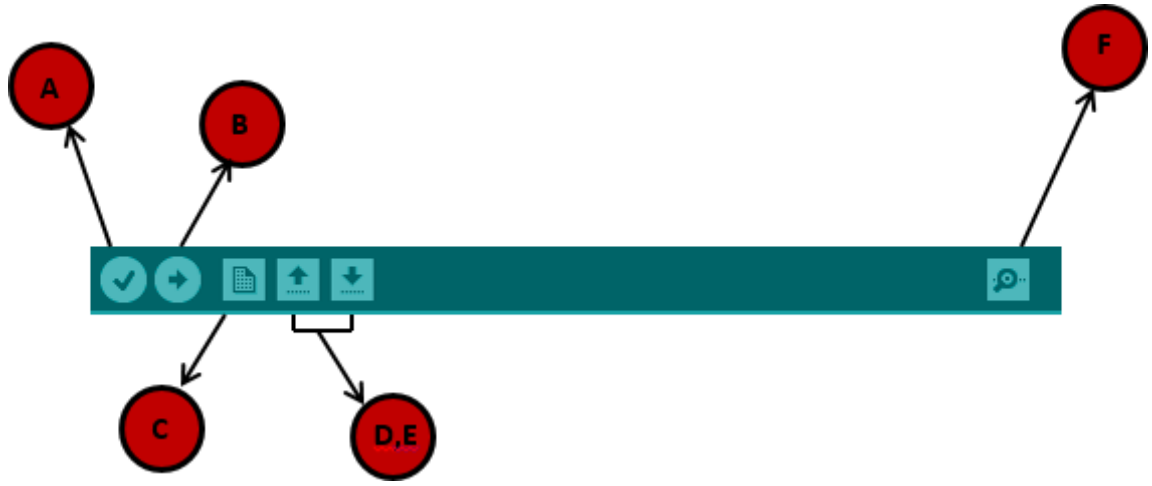
Step 7 – Select your serial port.

Select the serial device of the Arduino board. Go to **Tools → Serial Port** menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.



Step 8 – Upload the program to your board.

Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar.



A − Used to check if there is any compilation error.

B − Used to upload a program to the Arduino board.

C − Shortcut used to create a new sketch.

D − Used to directly open one of the example sketch.

E − Used to save your sketch.

F − Serial monitor used to receive serial data from the board and send the serial data to the board.

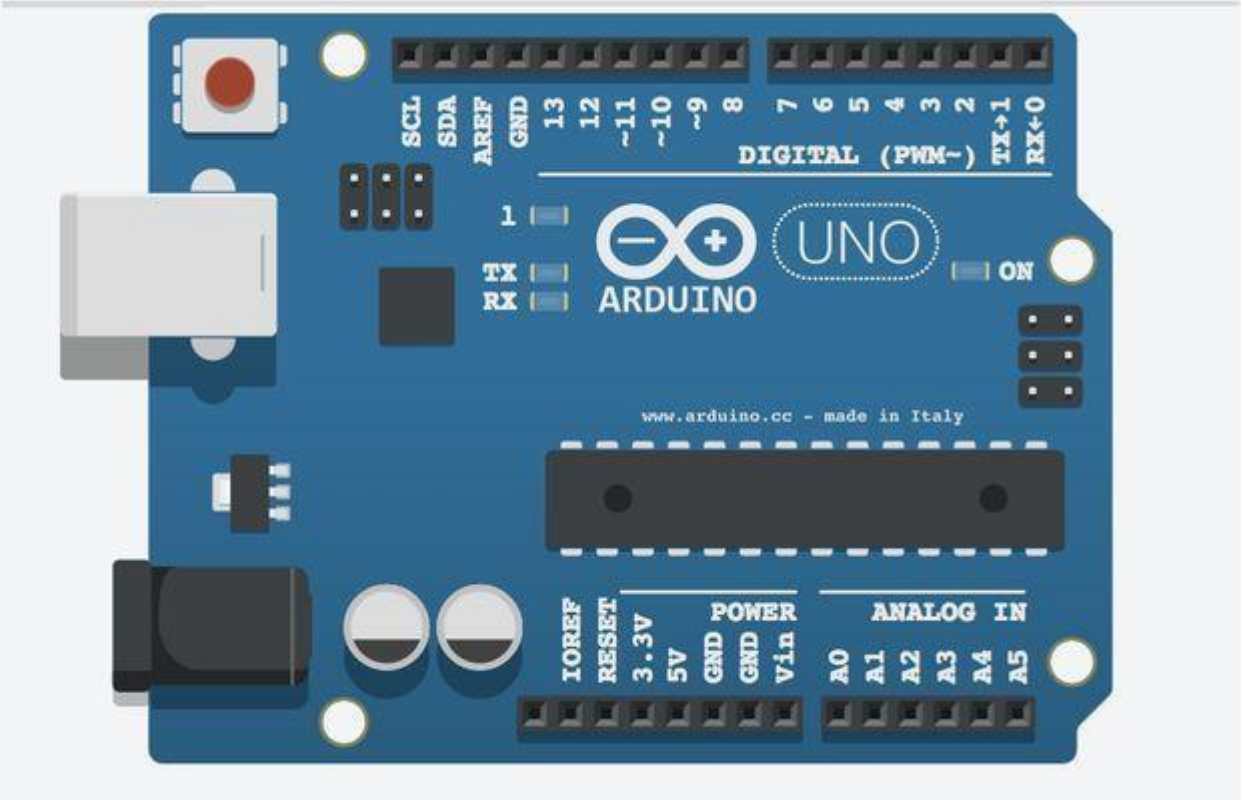
Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

IoT Platforms Overview: Arduino, Raspberry Pi

The IoT concepts imply a creation of network of various devices interacting with each other and with their environment. Interoperability and connectivity wouldn't be possible without hardware platforms that help developers solve issues such as building autonomous interactive objects or completing common infrastructure related tasks.

Let's go through the most popular IoT platforms and see how they work and benefit IoT software developers.

Arduino



The Arduino platform was created back in 2005 by the Arduino company and allows for open source prototyping and flexible software development and back-end deployment while providing significant ease of use to developers, even those with very little experience building IoT solutions.

Arduino is sensible to literally every environment by receiving source data from different external sensors and is capable to interact with other control elements over various devices, engines and drives. Arduino has a built-in micro controller that operates on the Arduino software.

Projects based on this platform can be both standalone and collaborative, i.e. realized with use of external tools and plugins. The integrated development environment (IDE) is composed of the open source code and works equally good with Mac, Linux and Windows OS. Based on a *processing* programming language, the Arduino platform seems to be created for new users and for experiments. The processing language is dedicated to visualizing and building interactive apps using animation and Java Virtual Machine (JVM) platform.

Let's note that this programming language was developed for the purpose of learning basic computer programming in a visual context. It is an absolutely free project available to every interested person. Normally, all the apps are programmed in C/C++, and are wrapped with *avr-gcc* (WinAVR in OS Windows).

Arduino offers analogue-to-digital input with a possibility of connecting light, temperature or sound sensor modules. Such sensors as *SPI* or *I2C* may also be used to cover up to 99% of these apps' market.

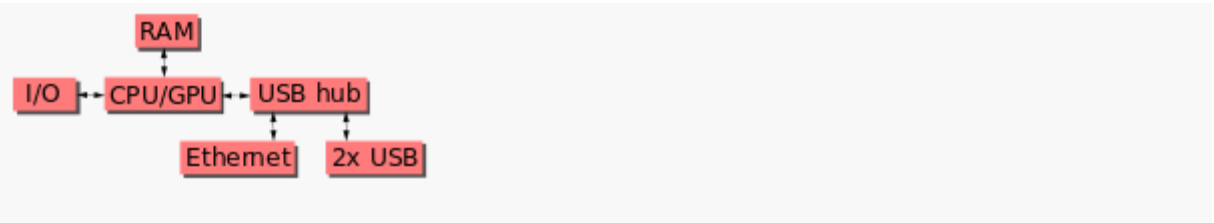
Arduino is a microcontroller (generally it is the 8-bit ATmega microcontroller), but not a mini-computer, which makes Arduino somehow limited in its features for advanced users. Arduino provides an excellent interactivity with external devices and offers a wide range of user manuals, project samples as well as a large community of users to learn from / share knowledge with.

Raspberry Pi

Raspberry Pi (/paɪ/) is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation in association with Broadcom. Early on, the Raspberry Pi project leaned towards the promotion of teaching basic computer science in schools and in developing countries. Later, the original model became far more popular than anticipated,selling outside its target market for uses such as robotics. It is now widely used in many areas, such as for weather monitoring,because of its low cost, modularity, and open design.

After the release of the second board type, the Raspberry Pi Foundation set up a new entity, named Raspberry Pi Trading, and installed Eben Upton as CEO, with the responsibility of developing technology.The Foundation was rededicated as an educational charity for promoting the teaching of basic computer science in schools and developing countries.The Raspberry Pi is one of the best-selling British computers.

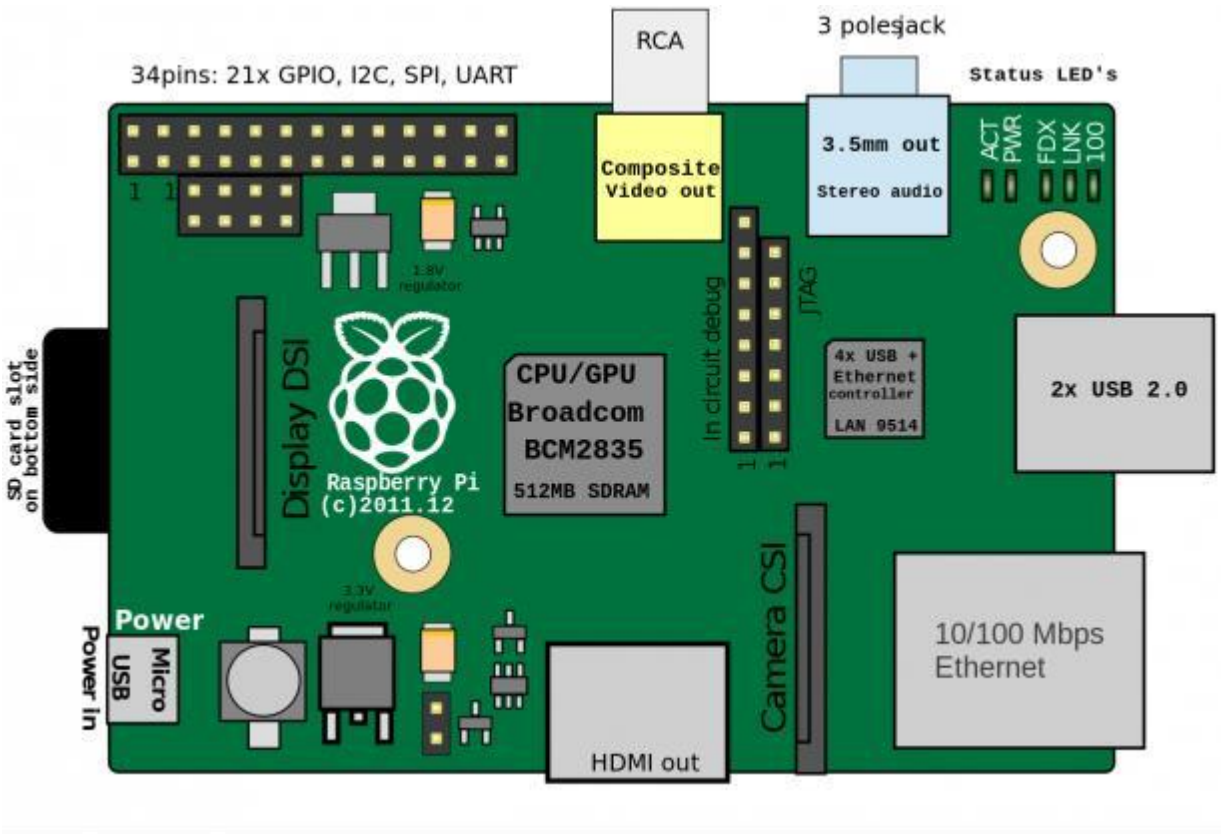
The Raspberry Pi hardware has evolved through several versions that feature variations in the type of the central processing unit, amount of memory capacity, networking support, and peripheral-device support.



This block diagram describes Model B and B+; Model A, A+, and the Pi Zero are similar, but lack the Ethernet and USB hub components. The Ethernet adapter is internally connected to an additional USB port. In Model A, A+, and the Pi Zero, the USB port is connected directly to the system on a chip (SoC). On the Pi 1 Model B+ and later models the USB/Ethernet chip contains a five-port USB hub, of which four ports are available, while the Pi 1 Model B only provides two. On the Pi Zero, the USB port is also connected directly to the SoC, but it uses a micro USB (OTG) port. Unlike all other Pi models, the 40 pin GPIO connector is omitted on the Pi Zero, with solderable through-holes only in the pin locations. The Pi Zero WH remedies this.

Processor speed ranges from 700 MHz to 1.4 GHz for the Pi 3 Model B+ or 1.5 GHz for the Pi 4; on-board memory ranges from 256 MiB to 1 GiB random-access memory (RAM), with up to 8 GiB available on the Pi 4. Secure Digital (SD) cards in MicroSDHC form factor (SDHC on early models) are used to store the operating system and program memory. The boards

have one to five USB ports. For video output, HDMI and composite video are supported, with a standard 3.5 mm tip-ring-sleeve jack for audio output. Lower-level output is provided by a number of GPIO pins, which support common protocols like I²C. The B-models have an 8P8C Ethernet port and the Pi 3, Pi 4 and Pi Zero W have on-board Wi-Fi 802.11n and Bluetooth.



Raspberry Pi is a mono-board computing platform that's as tiny as a credit card. Initially it was developed for computer science education with later on progress to wider functions.

Since the inception of Raspberry, the company sold out more than 8 million items. Raspberry Pi 3 is the latest version and it is the first 64-bit computing board that also comes with built-in Wi-Fi and Bluetooth functions. According to Raspberry Pi Foundation CEO Eben Upton, "*it's been a year in the making*". The Pi3 version is replaced with a quad-core 64-bit 1.2 GHz ARM Cortex A53 chip, 1GB of RAM, VideoCore IV graphics, Bluetooth 4.1 and 802.11n Wi-Fi. The developers claim the new architecture delivers an average 50% performance improvement over the Pi 2.

Another peculiarity of Raspberry Pi is the *GPIO* (General Purpose Input-Output), which is a low-level interface of self-operated control by input-output ports. Raspberry has it as a 40-pin connector.

Raspberry Pi uses Linux as its default operating system (OS). It's also fully Android compatible. Using the system on Windows OS is enabled through any virtualization system like *XenDesktop.* If you want to develop an application for Raspberry Pi on your computer, it

is necessary to download a specific toolset comprised of ARM-compiler and some libraries complied down to ARM-target platform like *glibc.*

DATA ANALYTICS AND SUPPORTING SERVICES

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

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

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

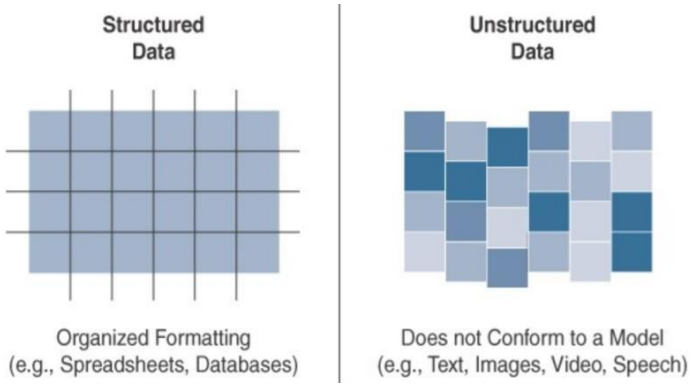
The potential for a petabyte (PB) of data per day per commercial airplane is not farfetched—and this is just for one airplane. Across the world, there are approximately 100,000 commercial flights per day. The amount of IoT data coming just from the commercial airline business is overwhelming. This example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

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

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

Structured vs unstructured data

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



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

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

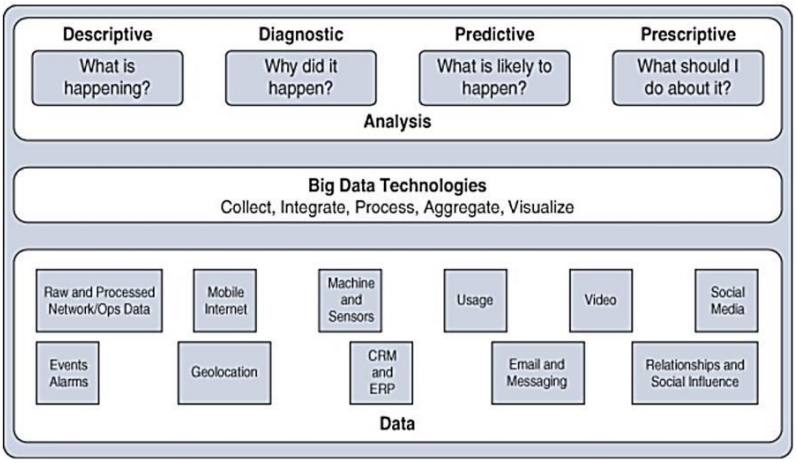
Smart objects in IoT networks generate both structured and unstructured data. Structured data is more easily managed and processed due to its well-defined organization. On the other hand, unstructured data can be harder to deal with and typically requires very different analytics tools for processing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

Data in motion versus data at rest

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

When data arrives at the data center, it is possible to process it in real-time, just like at the edge, while it is still in motion. Tools with this sort of capability, such as Spark, Storm, and Flink, are relatively nascent compared to the tools for analyzing stored data. Later sections of this chapter provide more information on these real-time streaming analysis tools that are part of the Hadoop ecosystem. Data at rest in IoT networks can be typically found in IoT brokers or in some sort of storage array at the data center. Myriad tools, especially tools for structured data in relational databases, are available from a data analytics perspective. The best known of these tools is Hadoop. Hadoop not only helps with data processing but also data storage. It is discussed in more detail later.

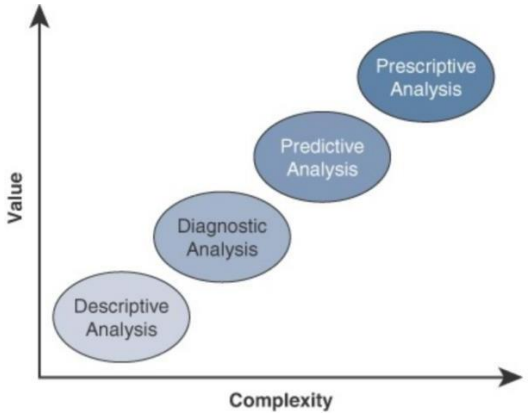
Descriptive: Descriptive data analysis tells you what is happening, either now or in the past. For example, a thermometer in a truck engine reports temperature values every second. From a descriptive analysis perspective, you can pull this data at any moment to gain insight into the current operating condition of the truck engine. If the temperature value is too high, then there may be a cooling problem or the engine may be experiencing too much load.



Diagnostic: When you are interested in the "why," diagnostic data analysis can provide the answer. Continuing with the example of the temperature sensor in the truck engine, you might wonder why the truck engine failed. Diagnostic analysis might show that the temperature of the engine was too high, and the engine overheated. Applying diagnostic analysis across the data generated by a wide range of smart objects can provide a clear picture of why a problem or an event occurred.

Predictive: Predictive analysis aims to foretell problems or issues before they occur. For example, with historical values of temperatures for the truck engine, predictive analysis could provide an estimate on the remaining life of certain components in the engine. These components could then be proactively replaced before failure occurs. Or perhaps if temperature values of the truck engine start to rise slowly over time, this could indicate the need for an oil change or some other sort of engine cooling maintenance.

Prescriptive: Prescriptive analysis goes a step beyond predictive and recommends solutions for upcoming problems. A prescriptive analysis of the temperature data from a truck engine might calculate various alternatives to cost-effectively maintain our truck. These calculations could range from the cost necessary for more frequent oil changes and cooling maintenance to installing new cooling equipment on the engine or upgrading to a lease on a model with a more powerful engine. Prescriptive analysis looks at a variety of factors and makes the appropriate recommendation.



IoT Data Analytics Challenges

As IoT has grown and evolved, it has become clear that traditional data analytics solutions were not always adequate. For example, traditional data analytics typically employs a standard RDBMS and corresponding tools, but the world of IoT is much more demanding. While relational databases are still used for certain data types and applications, they often struggle with the nature of IoT data. IoT data places two specific challenges on a relational database:

Scaling problems: Due to the large number of smart objects in most IoT networks that continually send data, relational databases can grow incredibly large very quickly. This can result in performance issues that can be costly to resolve, often requiring more hardware and architecture changes.

Volatility of data: With relational databases, it is critical that the schema be designed correctly from the beginning. Changing it later can slow or stop the database from operating. Due to the lack of flexibility, revisions to the schema must be kept at a minimum. IoT data, however, is volatile in the sense that the data model is likely to change and evolve over time. A dynamic schema is often required so that data model changes can be made daily or even hourly.

To deal with challenges like scaling and data volatility, a different type of database, known as NoSQL, is being used. Structured Query Language (SQL) is the computer language used to communicate with an RDBMS. As the name implies, a NoSQL database is a database that does not use SQL. It is not set up in the traditional tabular form of a relational database. NoSQL databases do not enforce a strict schema, and they support a complex, evolving data model. These databases are also inherently much more scalable.

In addition to the relational database challenges that IoT imposes, with its high volume of smart object data that frequently changes, IoT also brings challenges with the live streaming nature of its data and with managing data at the network level. Streaming data, which is generated as smart objects transmit data, is challenging because it is usually of a very high volume, and it is valuable only if it is possible to analyze and respond to it in real-time. Real-time analysis of streaming data allows you to detect patterns or anomalies that could indicate a problem or a situation that needs some kind of immediate response. To have a chance of affecting the outcome of this problem, you naturally must be able to filter and analyze the data while it is occurring, as close to the edge as possible. The market for analyzing streaming data in real-time is growing fast.

Major cloud analytics providers, such as Google, Microsoft, and IBM, have streaming analytics offerings, and various other applications can be used in house. (Edge streaming analytics is discussed in depth later in this chapter.) Another challenge that IoT brings to analytics is in the area of network data, which is referred to as network analytics. With the large numbers of smart objects in IoT networks that are

communicating and streaming data, it can be challenging to ensure that these data flows are effectively managed, monitored, and secure. Network analytics tools such as Flexible NetFlow and IPFIX provide the capability to detect irregular patterns or other problems in the flow of IoT data through a network. Network analytics, including both Flexible NetFlow and IPFIX, is covered in more detail.

Data acquiring

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

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

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

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

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

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

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

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

Message means a communicating entity or object.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Data Acquiring

1. Data Generation

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

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

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

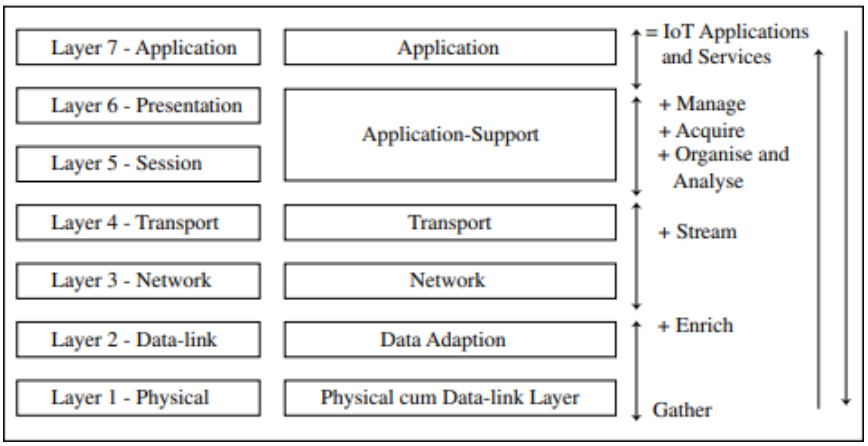
Event data: A device can generate data on an event only once. For example, on detection of the traffic or on dark ambient conditions, which signals the event. The event on darkness communicates a need for lighting up a group of streetlights (Example 1.2). A system consisting of security cameras can generate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

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

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

2. Data acquisition

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



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

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

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

3. Data validation

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

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

4. Data Categorization for Storage

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

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

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

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

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

5. Assembly Software for the Events

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

6. Data store

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

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

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

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

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

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

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

7. Data Centre Management

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

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

8. Server Management

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

● Short reaction times when the system or network is down

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

● Optimised performance

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

● Security of systems and protection

● Maintaining confidentiality and privacy of data

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

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

● Strict documentation and audit of all activities.

9. Spatial Storage

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

server

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

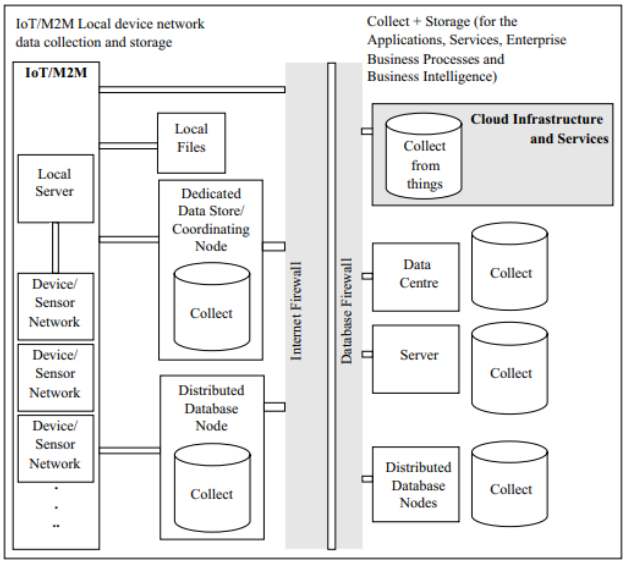
Cloud is a new generation method for data collection, storage and computing. cloud computing paradigm for data collection, storage, computing and services. describes cloud-computing service models in a software architectural concept, 'everything as a service'.Describes IoT specific cloud based services,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

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

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

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

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



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Cloud Computing Paradigm

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

Cloud Platform Services

Cloud platform offers the following:

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

automobiles and device networks

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

Cloud Platform Usages

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

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

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

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

Virtualization

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

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

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

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

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

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

Cloud Computing Features and Advantages

Essential features of cloud storage and computing are:

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

● Resource pooling in multi-tenant model

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

● Elasticity

● Massive scale availability

● Scalability

● Maintainability

● Homogeneity

● Virtualisation

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

● Resilient computing

● Advanced security

● Low cost

Cloud Computing Concerns

Concerns in usage of cloud computing are:

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

● Possible data loss

● Non delivery as per defined SLA specified performance

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

Cloud Deployment Models

Following are the four cloud deployment models:

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

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

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

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

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

Everything as a service and cloud service models

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

Cloud computing can be considered by a simple equation:

Cloud Computing = SaaS + Paas + IaaS + DaaS

SaaS means Software as a Service. The software is made available to an application or service on demand. SaaS is a service model where the applications or services deploy and host at the cloud, and are made available through the Internet on demand by the service user. The

software control, maintenance, updation to new version and infrastructure, platform and resource requirements are the responsibilities of the cloud service provider.

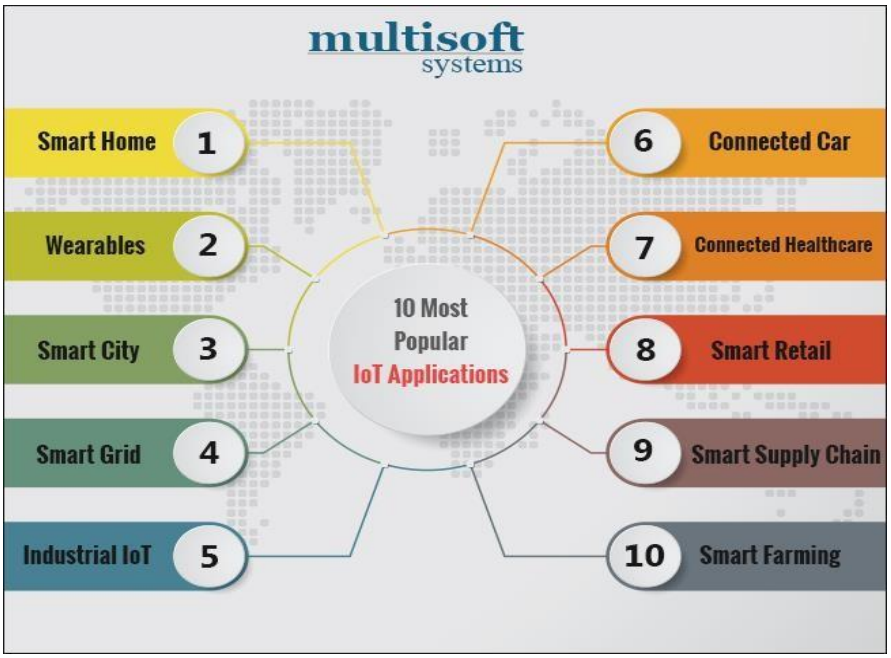
PaaS means Platform as a Service. The platform is made available to a developer of an application on demand. PaaS is a service model where the applications and services develop and execute using the platform (for computing, data store and distribution services) which is made available through the Internet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

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

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

Case Study

- **IoT applications in home, infrastructures, buildings, security, Industries, Home appliances, other IoT electronic equipments.**
- **Industry 4.0 concepts**



Case Study:

Definition:

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

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

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

IoT Applications:

IoT applications promise to bring immense value into our lives. With newer wireless networks, superior sensors and revolutionary computing capabilities, the **Internet of Things** could be the next frontier in the race for its share of the wallet. IoT applications are expected to equip billions of everyday objects with connectivity and intelligence. It is already being deployed extensively, few applications of IoT:

- Wearables
- Smart Home Applications
- Smart Buildings
- Smart Infrastructure
- Securities
- Health Care
- Smart Cities
- Agriculture
- Industrial Automation

IoT Applications:Smart Home, Smart Buildings and Infrastructure

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

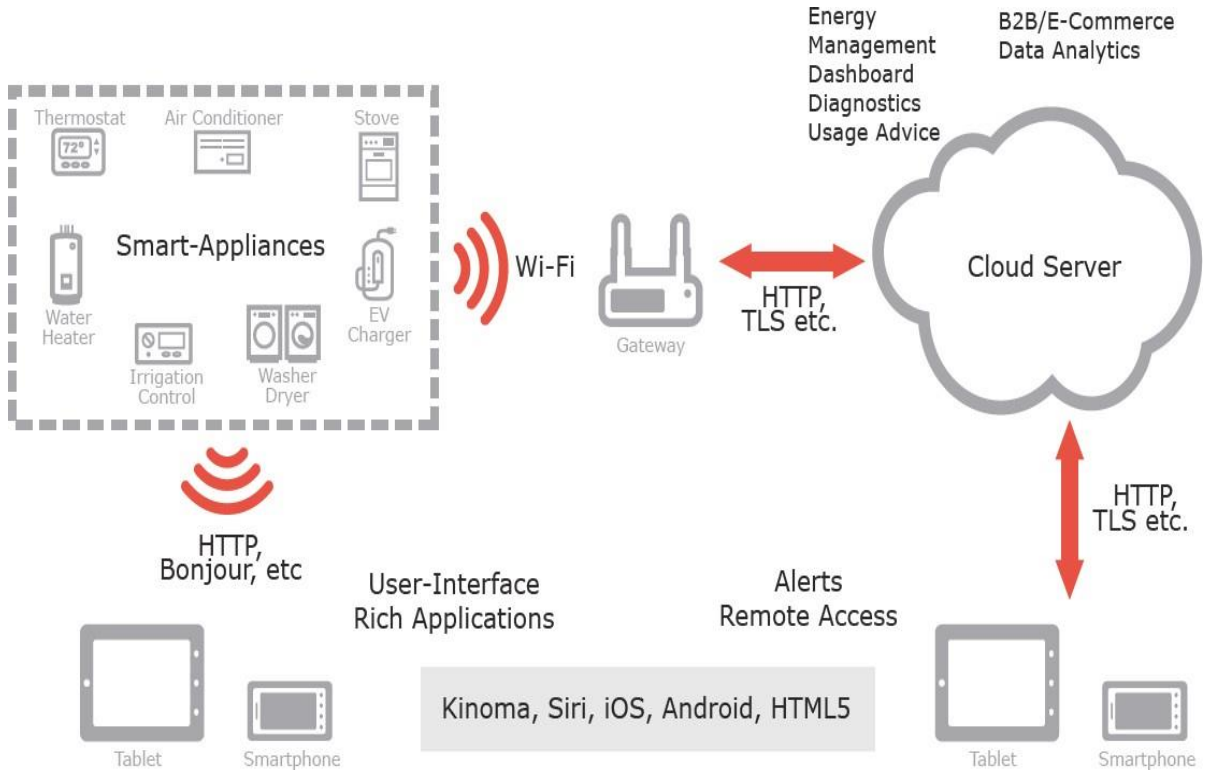


Figure .Smarthomeplatform.

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

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

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

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

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

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

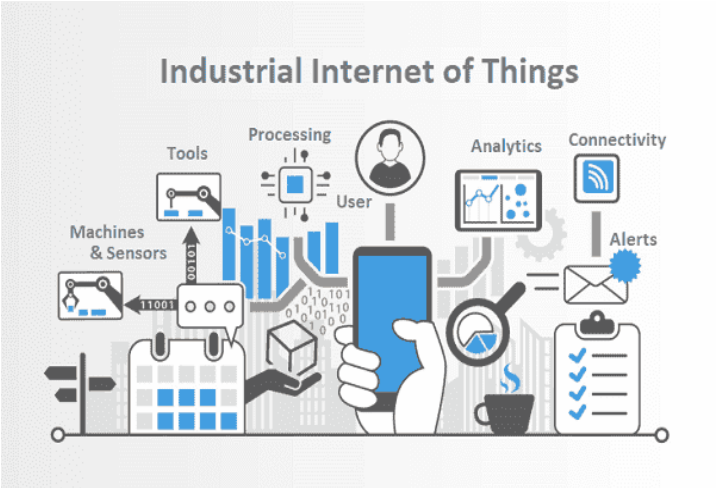
Within this field of research the exploitation of the potential of wirelesssensor networks (WSNs) to facilitate intelligent energy management in buildings,which increases occupant comfort while reducing energy demand, ishighly relevant.

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

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

IoT Application in industries:

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



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

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

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

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

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

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

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

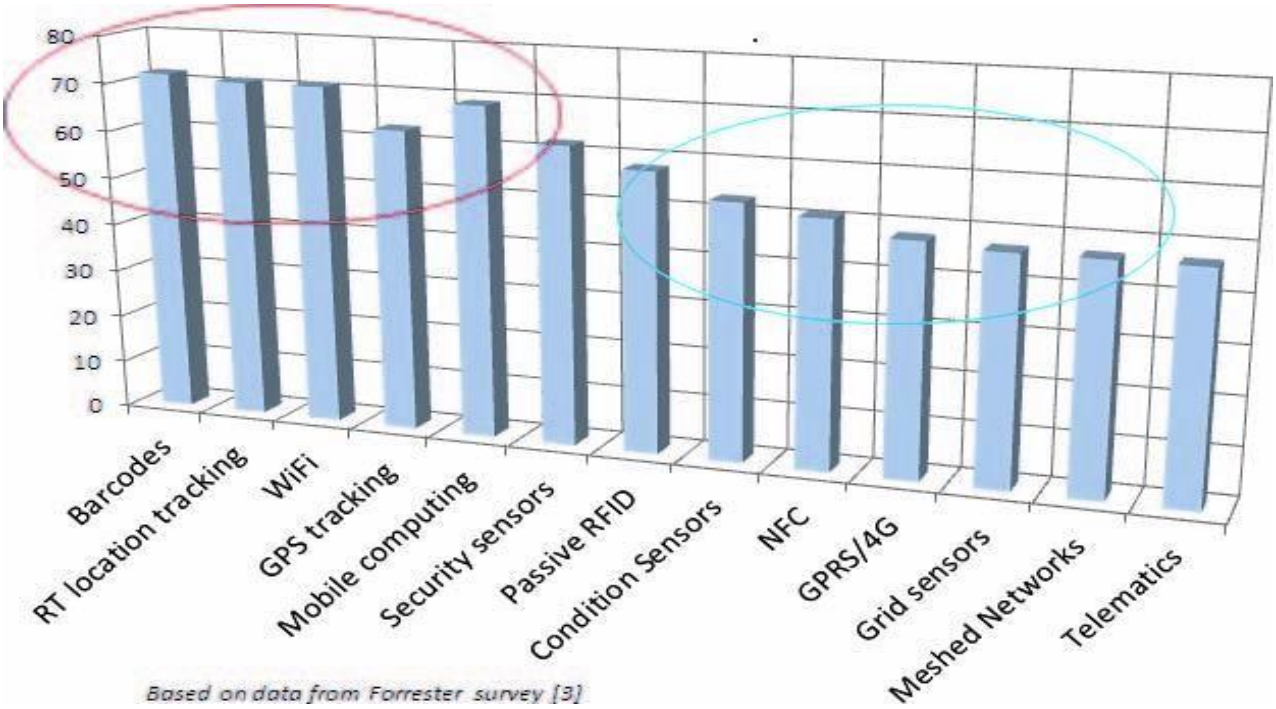


Fig. 5.2 View on very important and important perceived IoT technologies expected to bring value in applications.

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

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

Figure.Status and estimated potential of IoT applications.

IoT application requirements and capabilities:

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

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

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

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

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

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

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

Challenges faced by IoT industry applications

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

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

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

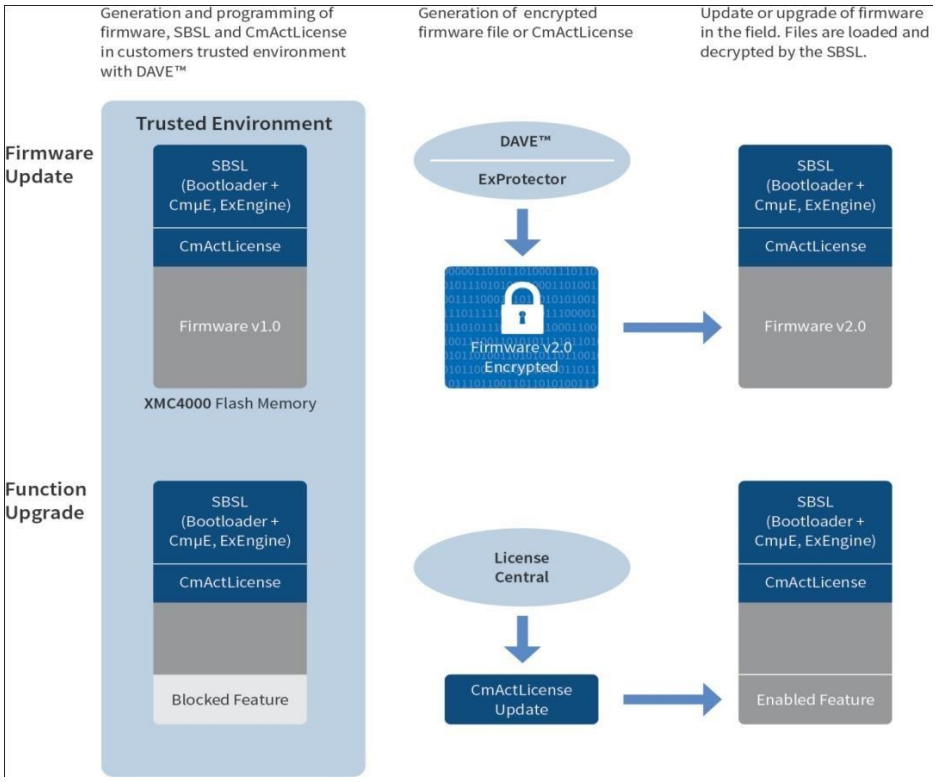
Seurity:

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

Privacy Risks:

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

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



IoT Application of home appliances:

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

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

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

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

Home Appliance in Internet of Things:

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

Smart Washing Machine:

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

Smart Refrigerator with Internet of Things:

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

Shortest Way to Dry Hair:

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

Smart Doorbell:

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

Smart Camera for Safe Home:

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

Industry 4.0 concepts

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

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

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

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

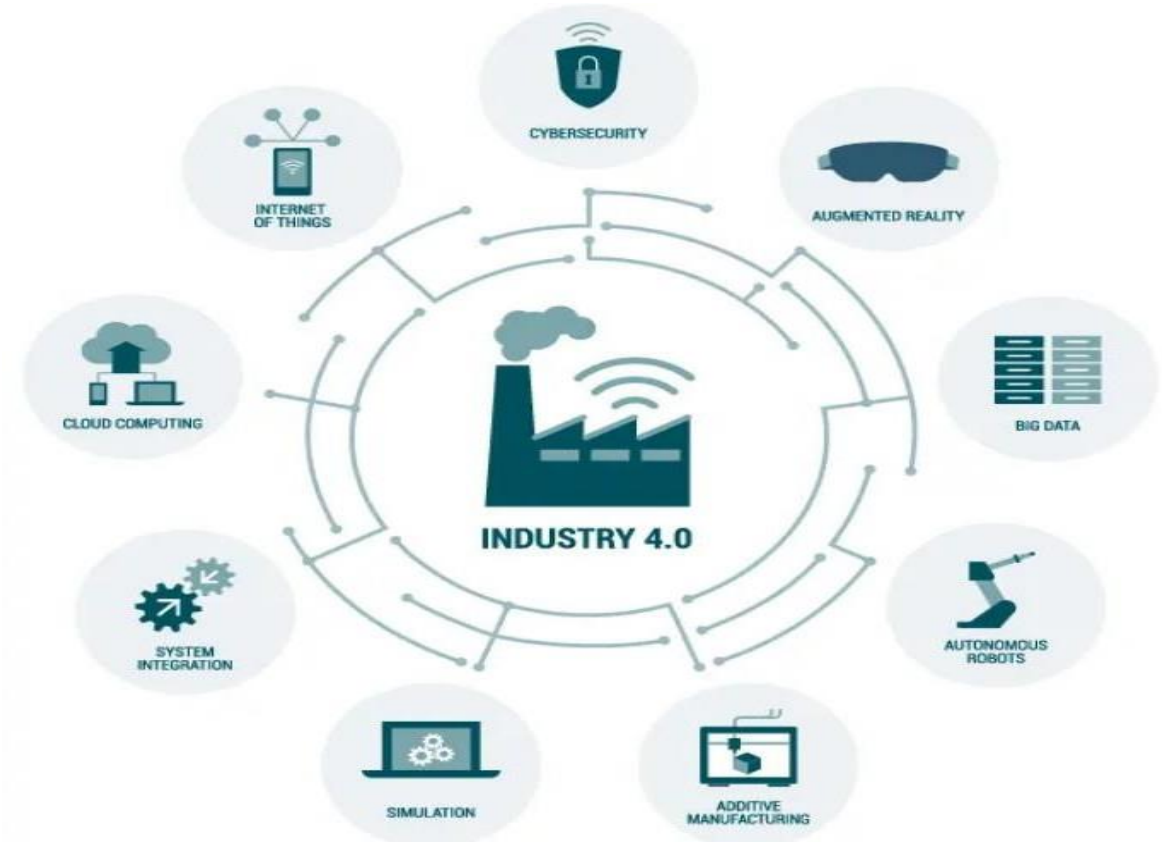
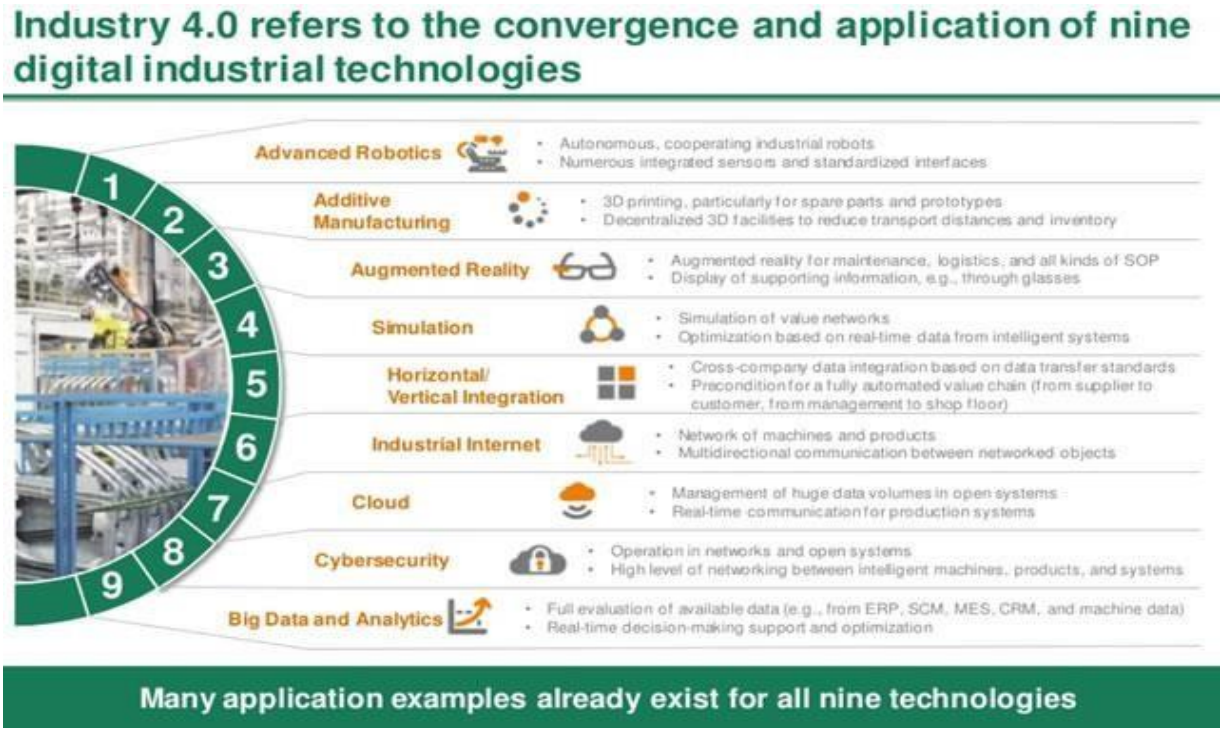


Figure: Nine Technologies of Industry 4.0



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

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

Evolution of Industry 4.0

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

1. The First Industrial Revolution

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

2. The Second Industrial Revolution

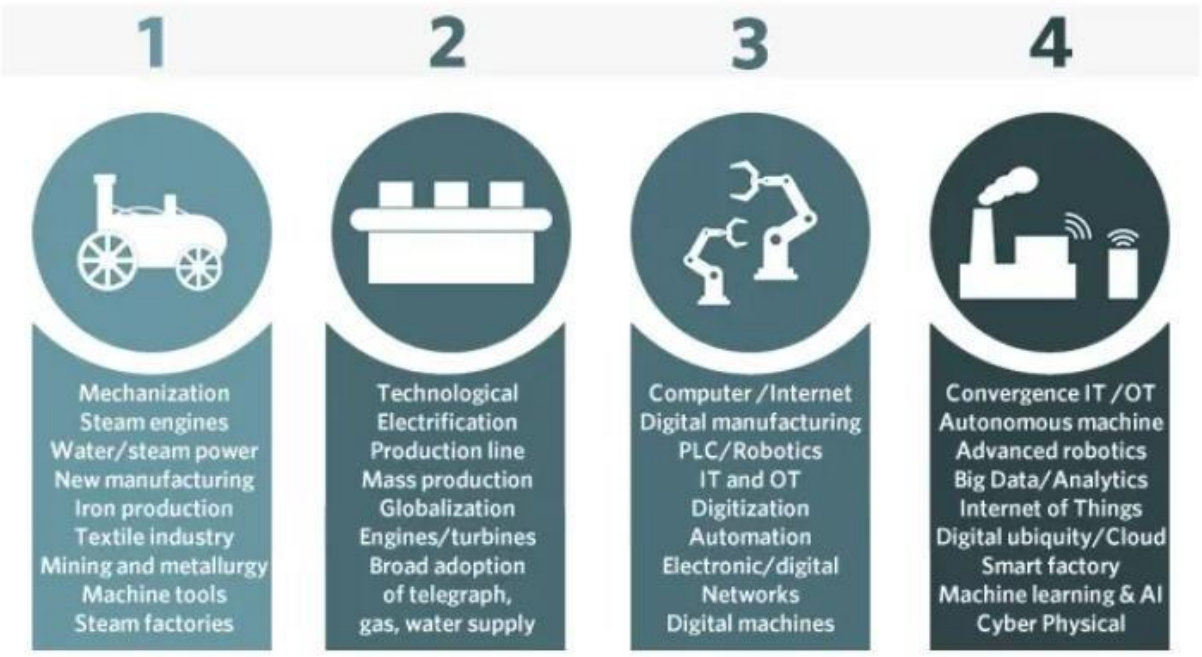
In the early part of the 20th century, the world entered a second industrial revolution with the introduction of steel and use of electricity in factories. The introduction of electricity enabled manufacturers to increase efficiency and helped make factory machinery more mobile. It was during this phase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

3. Third Industrial Revolution

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

4. Fourth Industrial Revolution[Industry 4.0]

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



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

1) even more automation than in the third industrial revolution

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

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

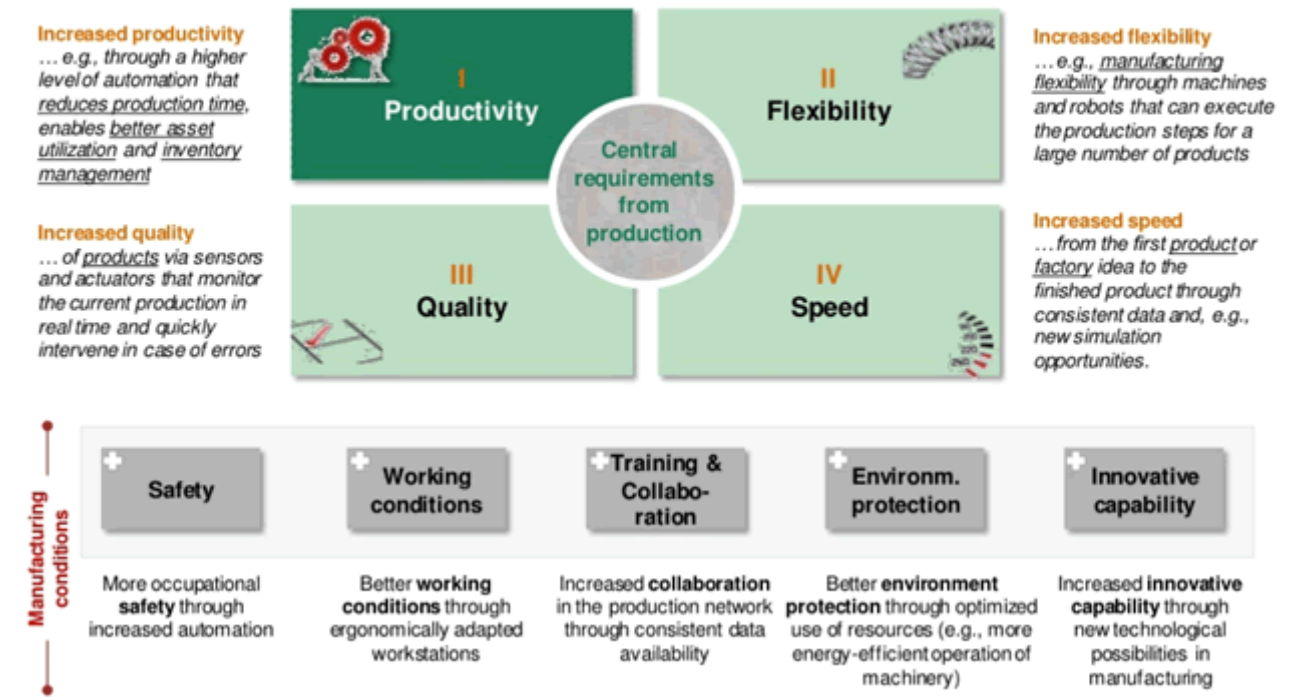
4) closed-loop data models and control systems and

5) personalization/customization of products.

Benefits of Industry 4.0

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

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



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

1. Enhanced productivity through optimization and automation

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

3. Higher business continuity through advanced maintenance and monitoring possibilities

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

5. Better working conditions and sustainability

6. Personalization and customization for the ‘new’ consumer

7. Improved agility

8. The development of innovative capabilities and new revenue models