# 1 Introduction to Internet of Things

UNIT- I

**Syllabus**

At the end of this unit, you should be able to understand and comprehend the following syllabus topics:

- IoT Definition
- IoT characteristics
- M2M and IoT
- End to End IoT Architecture
- Physical design of IoT
- Logical Design of IoT
- Overview of IoT protocols
- IoT levels and deployment templates
- Challenges for IoT
- Interdependencies of IoT and cloud computing
- Web of things

## 1.1 Introduction to Internet of Things (IoT)

I am sure you would have heard of several smart devices such as smart watches, smart bulbs, smart thermostat, smart tv, smart refrigerators, and perhaps anything that comes with a prefix of "smart". These devices are increasingly being used and serve various purposes such as health monitoring, regulating temperature, capturing video or audio intelligently, etc. These devices are also commonly called as Internet of Things or IoT in short.

✍ ***Definition :*** *The Internet of Things (IoT) is a collection of diverse technologies that interact with the physical world.*

In a nutshell, IoT refers to not only smart devices but also the underlying technologies that enable and connect these devices to the internet. There are billions of such devices in homes, schools, organisations, factories, oil wells, hospitals, cars, and several other places. With the explosion of these connected devices, you increasingly need solutions to connect them, and collect, store, and analyse device data to gain powerful and actionable insights.

The Fig. 1.1.1 shows the broad spectrum of common IoT devices used either directly by the consumer or in an industry setting.

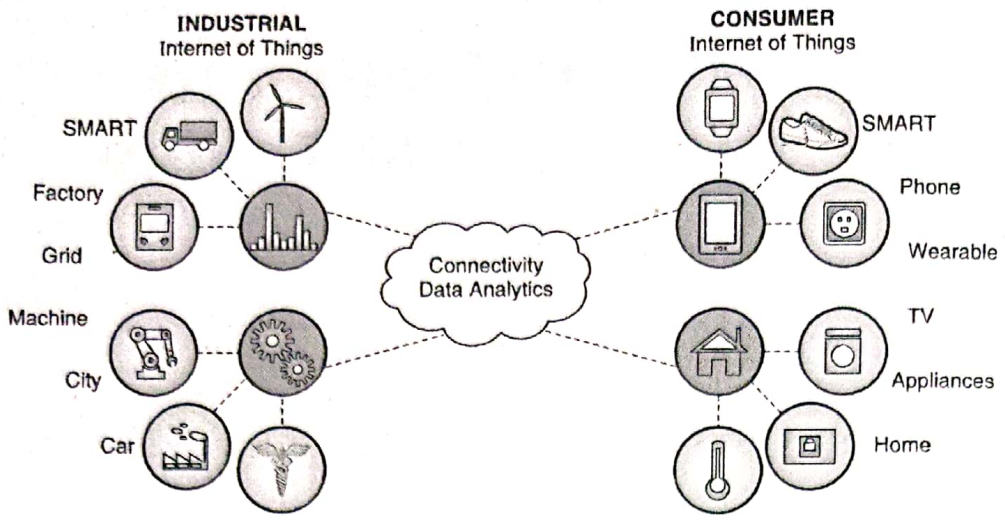



**Fig. 1.1.1**

In this section, you will learn about several technologies, protocols and architectural solutions that make IoT possible.

### 1.1.1 Characteristics of IoT

The major characteristics of IoT are as shown in Fig. 1.1.2.

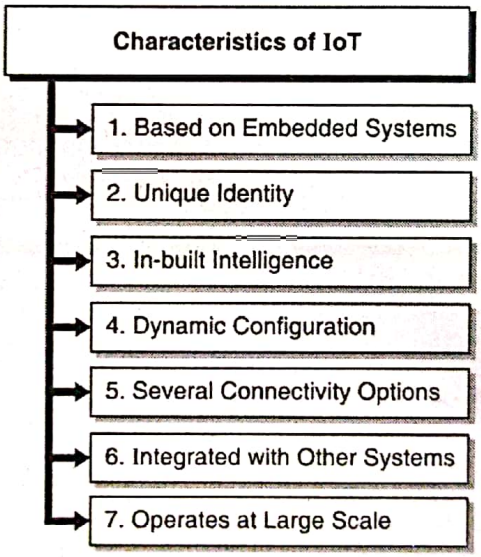



**Fig. 1.1.2**

1. **Based on Embedded Systems :** IoT devices are typically based on embedded systems and inherit several of their characteristics such as real-timeliness, low power consumption, high reliability, tolerance for harsh environments and small form factor. These devices perform specific functions such as reading temperature or controlling lights. They typically run specific Operating Systems (OS) as appropriate for the task they need to carry out.
2. **Unique Identity :** You need to control, operate, and manage IoT devices remotely via various connectivity options such as Wi-Fi and Bluetooth. Hence, these devices have unique identities such as IP addresses, MAC addresses, serial numbers, etc. using which you can precisely address a specific device and can either read output from the device (say temperature reading) or provide input (to turn off or turn on) to the device.

3. **In-built Intelligence :** As you understand that IoT devices are built on top of embedded systems, these devices are packed with in-built intelligence in the form of specific algorithms, situation-to-action mapping, and other intelligent mechanisms that can sense the environmental conditions and take specific actions or provide the desired response. For example, a smart bulb could automatically detect when you enter and leave the room and power on or off accordingly.

4. **Dynamic Configuration :** IoT devices are self-adapting and self-configuring in nature where they can appropriately and dynamically reconfigure themselves depending upon the operating environment, situations, and conditions. Their operation could be automated where you need not provide input controls to get the desired behaviour.

   For example, an automatic watering system can water the plants based on the soil moisture. On sunny days, the soil might dry up quickly leading to frequent watering and on a non-sunny day, it might only require watering twice a day. How fast the soil loses moisture depends upon a lot of external conditions and it is hard to predict the precise timing when watering would be required. Such systems can completely work autonomously where you could just set the desired soil moisture to retain and the system can continuously monitor the soil moisture and adjust watering accordingly. You don't really need to tell how many times to switch on the watering system.

   Another example could be CCTV cameras that could send low resolution images when there is no movement near them to save network bandwidth but could automatically start to send high resolution images when they sense movement around them.

5. **Several Connectivity Options :** IoT devices typically support various network connectivity options that you could use as per your requirements. Some of the common connectivity options are 2G, 3G, 4G, LTE, 5G, Wi-Fi, Bluetooth, NFC, ZigBee, USB, Ethernet, etc. These devices also support a wide range of network communication protocols such as HTTP, CoAP, MQTT, AMQP, etc. When buying IoT devices, you should determine what connectivity options are available on the devices and make a judicious decision based on your requirements. All devices may not support all the connectivity options and network protocols at the same time. You should carefully choose what you need.

6. **Integrated with Other Systems :** IoT systems often work with other systems taking an integrated approach. IoT systems could provide the desired data that could make rest of the integrated systems work together. For example, you could have an IoT based fire alarm that could automatically use the telephone system to call fire station and other emergency responders, send the building evacuation orders to public announcement system, could automatically call a few ambulances and doctors to handle casualties, and most importantly could activate water sprinkler system to extinguish the fire. IoT devices often work in groups and interact with each other to achieve the desired outcome.

7. **Operates at large scale :** IoT devices operate at large scale. Consider a shopping mall or a factory. The total number of sensors could be in thousands. Overall, an organisation may be controlling, managing, and operating tens of thousands of devices that co-operate and carry out the desired activities.

   Also, the total number of IoT devices could easily surpass the human population and any other forms of computing. As per the Gartner report published on Aug 2019 (https://www.gartner.com/en/newsroom/press-releases/2019-08-29-gartner-says-5-8-billion-enterprise-and-automotive-io), approximately 5.8 Billion Enterprise and Automotive IoT Endpoints would be in use by end of 2020. The Table 1.1.1 from the report summarises the IoT endpoint market by segment, 2018-2020, worldwide (installed base, billions of units).

**Table 1.1.1**

| Segment | 2018 | 2019 | 2020 |
|---|---|---|---|
| Utilities | 0.98 | 1.17 | 1.37 |
| Government | 0.40 | 0.53 | 0.70 |
| Building Automation | 0.23 | 0.31 | 0.44 |
| Physical Security | 0.83 | 0.95 | 1.09 |
| Manufacturing & Natural Resources | 0.33 | 0.40 | 0.49 |
| Automotive | 0.27 | 0.36 | 0.47 |
| Healthcare Providers | 0.21 | 0.28 | 0.36 |
| Retail & Wholesale Trade | 0.29 | 0.36 | 0.44 |
| Information | 0.37 | 0.37 | 0.37 |
| Transportation | 0.06 | 0.07 | 0.08 |
| **Total** | **3.96** | **4.81** | **5.81** |

## 1.2 IoT Vision and Applications

IoT is expected to make a positive difference to multiple sectors – retail, cities, offices, homes, and perhaps everything around you. Let's understand some of the key impacts that IoT could have. Most of this section builds on the report, published by McKinsey & Company, titled "The Internet Of Things: Mapping The Value Beyond The Hype".

1. **Human :** For human-beings, the IoT applications fall into two broad categories – improving health and raising productivity. Unlike other IoT applications, where a reading from a sensor might initiate a specific action, in the human setting, sensor data provide information that people will use to guide their actions and decisions. Using IoT systems to convince healthy people to change their living habits and to help sick patients adhere to doctors' prescriptions would be a true breakthrough. Emerging applications have the potential to transform a wide range of health-care therapies. Implantables, ingestibles and injectables, such as smart pills and nanobots, have the potential eventually to replace many surgeries with less invasive procedures that could offer faster recovery, reduced risk of complications, and lower cost. Imagine an IoT device, placed into a human body, that can continuously monitor blood glucose level and automatically inject insulin dosage appropriately.
2. **Home :** IoT applications relating to the operation of homes, such as energy management, security, and automation of domestic chores are the most useful. By far, the largest opportunity in the home setting is in automating domestic chores. This work is not counted in national productivity data but has an enormous impact on how people spend time and money. The household activities such as cleaning, washing, preparing food, gardening, caring for pets, and so on take a good amount of time and effort.

   You could automatically have the home temperature controlled, house cleaned by a robotic vacuum cleaner, refrigerators or shelves automatically ordering groceries based on what is placed on them, clothes washed, dried, and ironed and several other home chores that could be automated.

**Various IoT applications in homes**

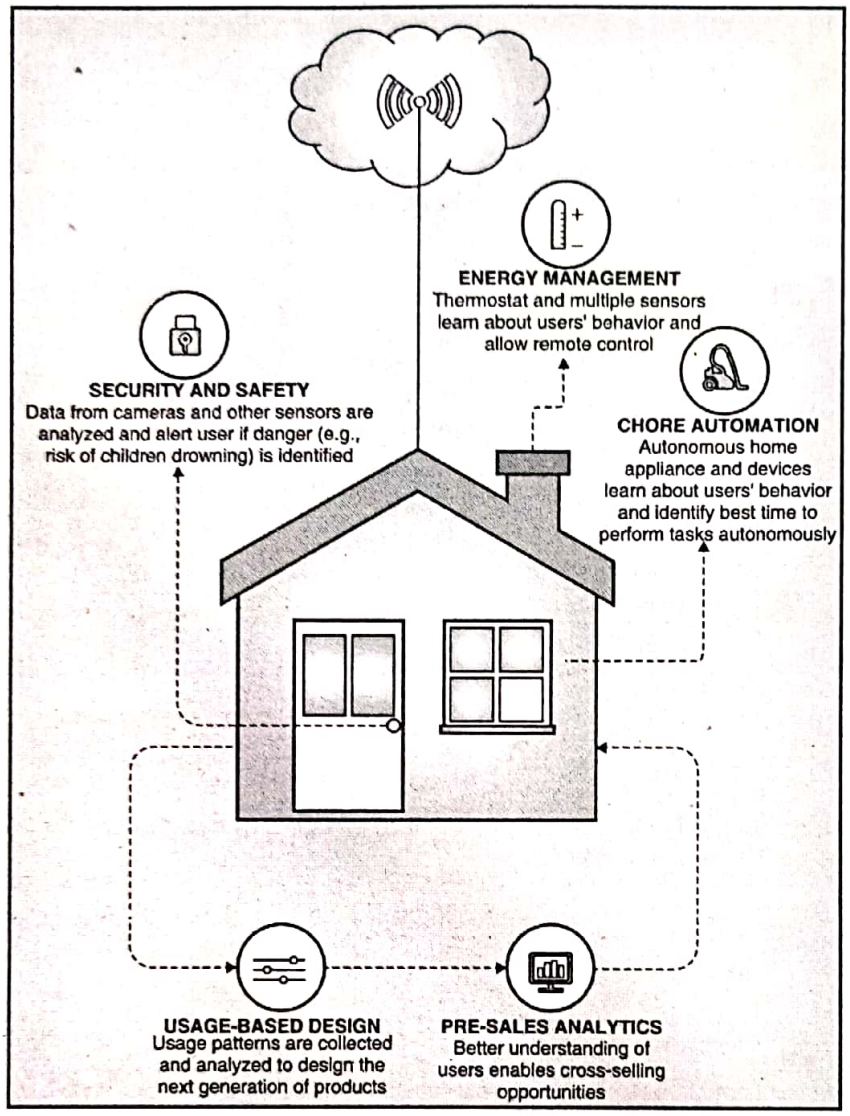



**Fig. 1.2.1 : Various IoT applications in homes**

3. **Retail :** Retail environments have undergone significant change over the past two decades due to the introduction of information technologies, including the rise of online shopping. The Internet of Things has the potential to cause even greater disruption, but IoT can also provide traditional retailers with the tools to compete and coexist with the online retail world as "omni-channel" shopping erases the distinction between online and offline shops. The Internet of Things, for example, can guide the shopper to the item she has been looking at online when she enters the store and text her a personalised coupon to make the purchase in-store that day. IoT technology can also provide data to optimise store layouts, enable fully automated checkout, and fine-tune inventory management. These and other innovations could enable new business models and allow retailers to improve productivity, reduce costs, and raise sales.

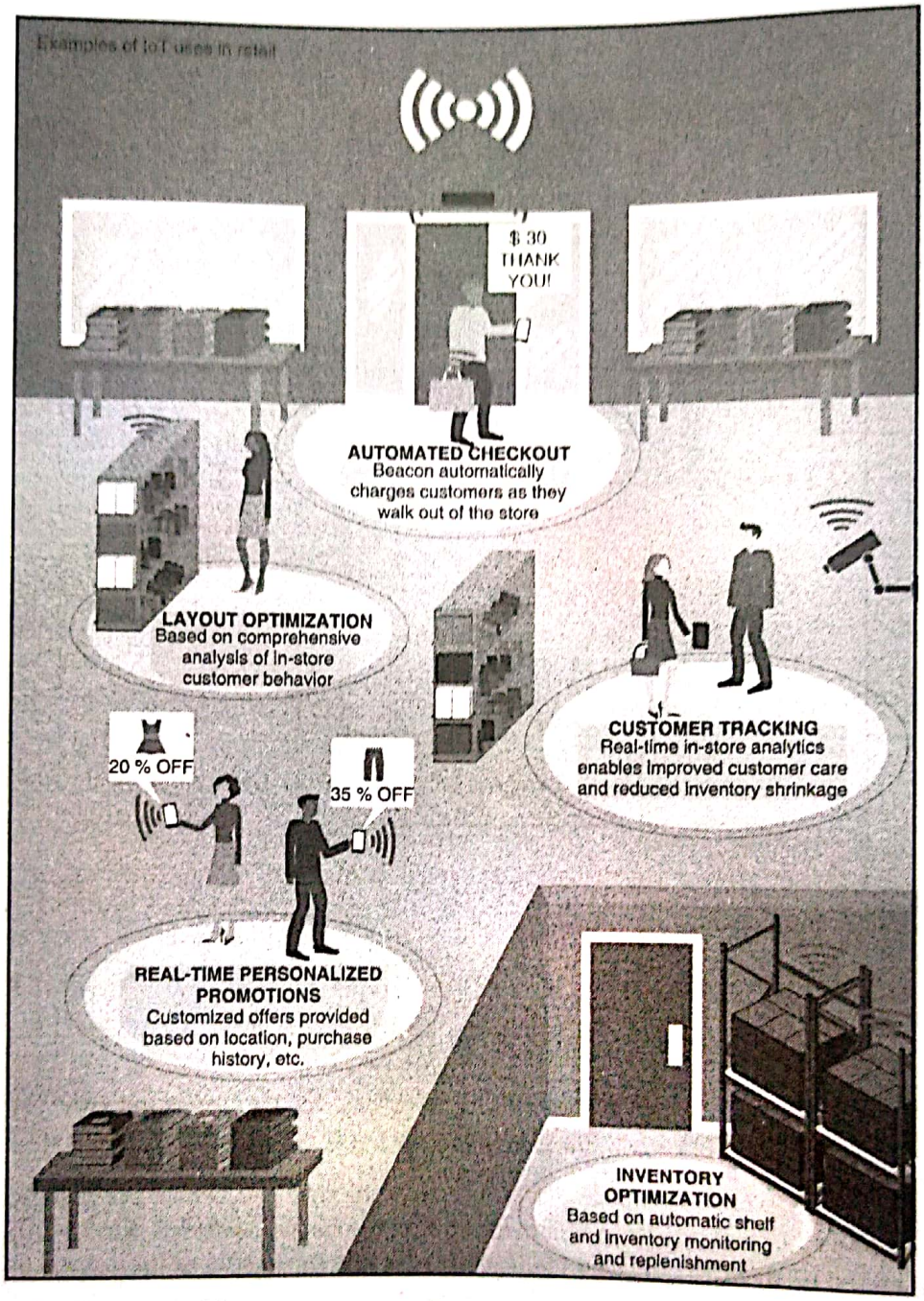



**Fig. 1.2.2 : Examples of IoT uses in retail**

Amazon Go is an interesting example of how IoT has drastically changed retail shopping experience. Amazon Go is an IoT enabled retail store that has demonstrated that having a cash counter and checkout system for billing could be a thing of the past. You could just pick the items from the store and walkout. Based on what you picked up, the bill is automatically generated and charged to your credit card or bank account. You can read more about it at https://www.amazon.com/b?ie=UTF8&node=16008589011. Here is a snapshot of frequently asked questions around Amazon Go.

## FREQUENTLY ASKED QUESTIONS

**What is Amazon Go?**

Amazon Go is a new kind of store with no checkout required. We created the world's most advanced shopping technology so you never have to wait in line. With our Just Walk Out Shopping experience, simply use the Amazon Go app to enter the store, take the products you want, and go! No lines, no checkout. (No, seriously.)

**How does Amazon Go work?**

Our checkout-free shopping experience is made possible by the same types of technologies used in self-driving cars: computer vision, sensor fusion, and deep learning. Our Just Walk Out Technology automatically detects when products are taken from or returned to the shelves and keeps track of them in a virtual cart. When you're done shopping, you can just leave the store. A little later, we'll send you a receipt and charge your Amazon account.

**What can I buy at Amazon Go?**

We offer delicious ready-to-eat breakfast, lunch, dinner, and snack options made by our chefs and favorite local kitchens and bakeries. Our selection of grocery essentials ranges from staples like bread and milk to artisan cheeses and locally made chocolates. For a quick home-cooked dinner, pick up one of our chef-designed Amazon Meal Kits, with all the ingredients you need to make a meal for two in about 30 minutes.

**How do I find out what products are in a specific Amazon Go store?**

To find out what a specific store sells, tap the Discover tab in the Amazon Go app.

**How do I shop at Amazon Go?**

All you need is an Amazon account, the free Amazon Go app, and a recent-generation iPhone or Android phone. You can find the Amazon Go app on the Apple App Store, Google Play, and Amazon Appstore.

When you arrive, use the app to enter the store, then feel free to put your phone away—you don't need it to shop. Then just browse and shop like you would at any other store. Once you're done shopping, you're on your way. No lines, no checkout.

**Do you have any people working in the store?**

Yes. Our great team of associates works in both the kitchen and the store to prep ingredients, make our ready-to-eat food, stock shelves, and help customers. (Need a product recommendation? Ask an associate!)

**Why did you build Amazon Go?**

We asked ourselves: what if we could create a shopping experience with no lines and no checkout? Could we push the boundaries of computer vision and machine learning to create a store where customers could simply take what they want and go? Our answer to those questions is Amazon Go and Just Walk Out Shopping.

**What if I have other questions about shopping at Amazon Go?**

You can find more information about the store in the Amazon Go app, including a tutorial on how to shop the store. After opening the Amazon Go app, tap More > Help.

4. **Offices** : IoT has applications in the office setting that are similar to those in the home setting such as managing energy and security systems. Quite a few office spaces are conserving natural resources by regulating the usage of electricity, water, garbage disposal, etc. and becoming "green" buildings.
5. **Factories :** IoT would play a major role in the next phase of factory automation. It would help in full digitisation of production processes, monitor and control all tools of production, and use the data collected to improve productivity and quality. The greatest potential for creating value would be in operations optimisation by making the various processes within the factory more efficient. This includes using sensors, rather than human judgment (and human error), to adjust the performance of machinery. It also involves use of data from production machinery to adjust workflows. This is done by remotely tracking, monitoring, and adjusting machinery based on sensor data from different parts of the plant (and even across plants).

   After operations optimisation, the next most valuable applications of IoT in the factory setting are predictive maintenance and inventory optimisation. Predictive maintenance involves using sensors to monitor machinery continuously to avoid breakdowns and determine then maintenance will be required, rather than relying on regularly scheduled maintenance routines.

   IoT can improve inventory management by automatically restocking parts bins based on weight or height data recorded by sensors.

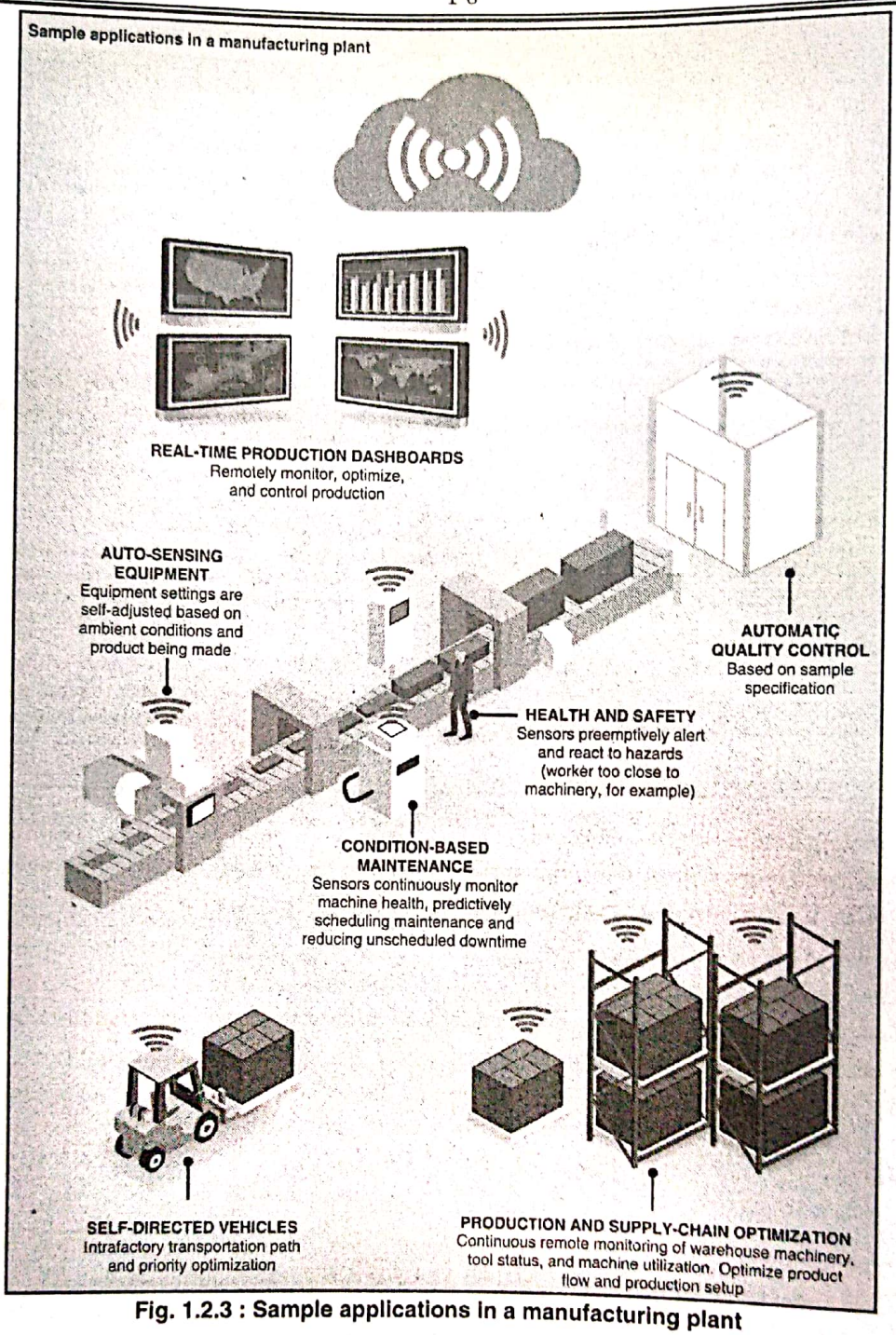



**Fig. 1.2.3 : Sample applications in a manufacturing plant**

6. **Vehicles :** IoT would play a major role in how vehicles such as cars, trains, trucks, even aircraft are used. The IoT sensors and connectivity could improve how vehicles are serviced, maintained, and designed. The same types of sensors and wireless connections that make it possible to create a self-driving car can be used to monitor how vehicles are performing to enable condition-based maintenance routines that are far more cost-effective than periodic maintenance or performing maintenance after a problem occurs. Tracking performance data can also help vehicle manufacturers design more reliable products and discover other ways to serve customers. Insurance companies could monitor your driving habits and patterns and could provide insurance quotation specifically for you instead of charging you fixed premiums.

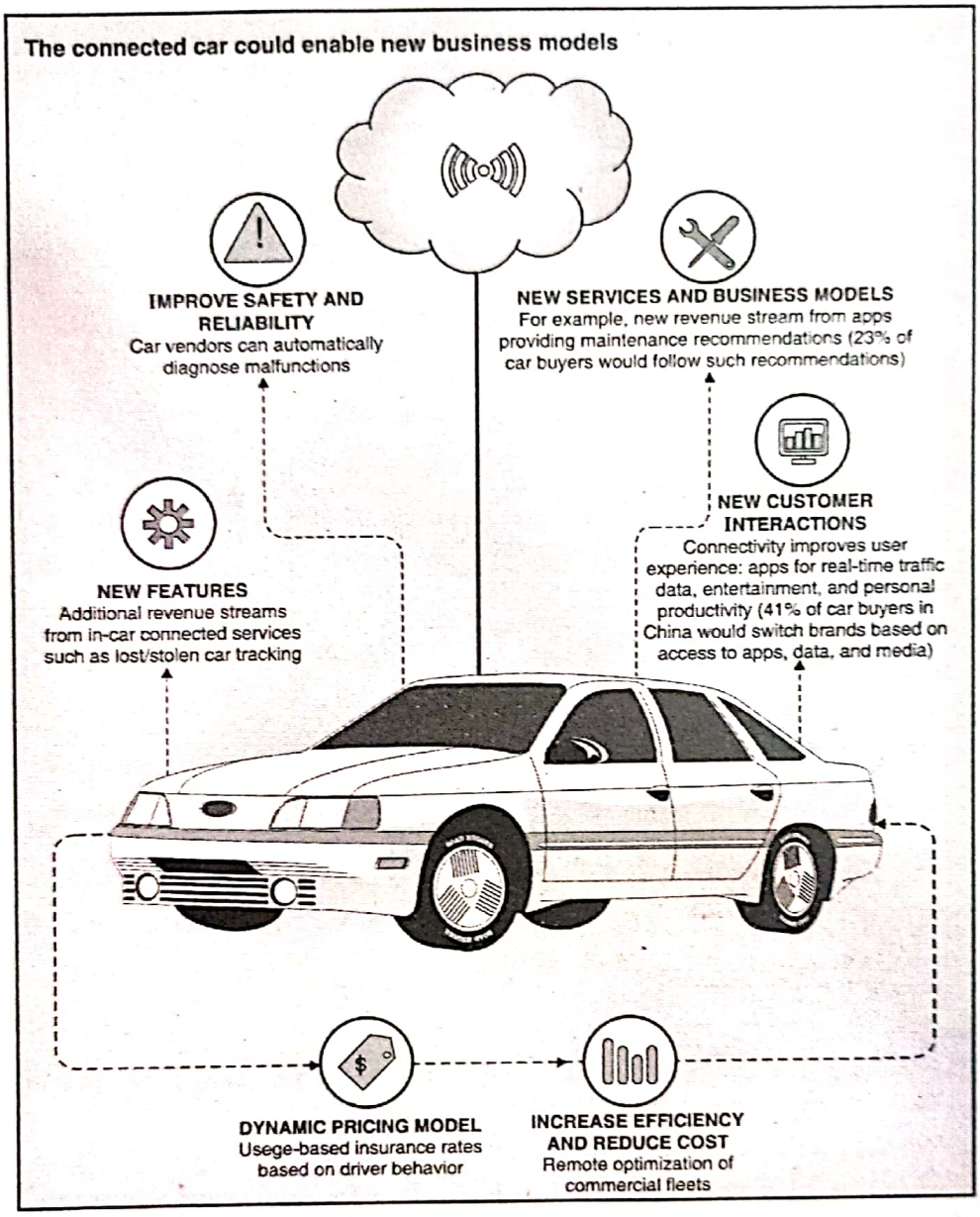



**Fig. 1.2.4 : Connected Car**

7. **Cities :** IoT has kicked off several "smart city" initiatives around the world. These cities are experimenting with IoT applications to improve services, relieve traffic congestion, conserve water and energy, and improve quality of life. Large, concentrated populations and complex infrastructure make cities a target-rich environment for IoT applications. IoT applications in cities are focusing on public health and safety, transportation, and resource management. IoT applications in public safety and health include air and water quality monitoring.

   Transportation applications range from traffic-control systems to smart parking meters to autonomous vehicles. Resource and infrastructure management uses include sensors and smart meters to better manage water and electric infrastructure.

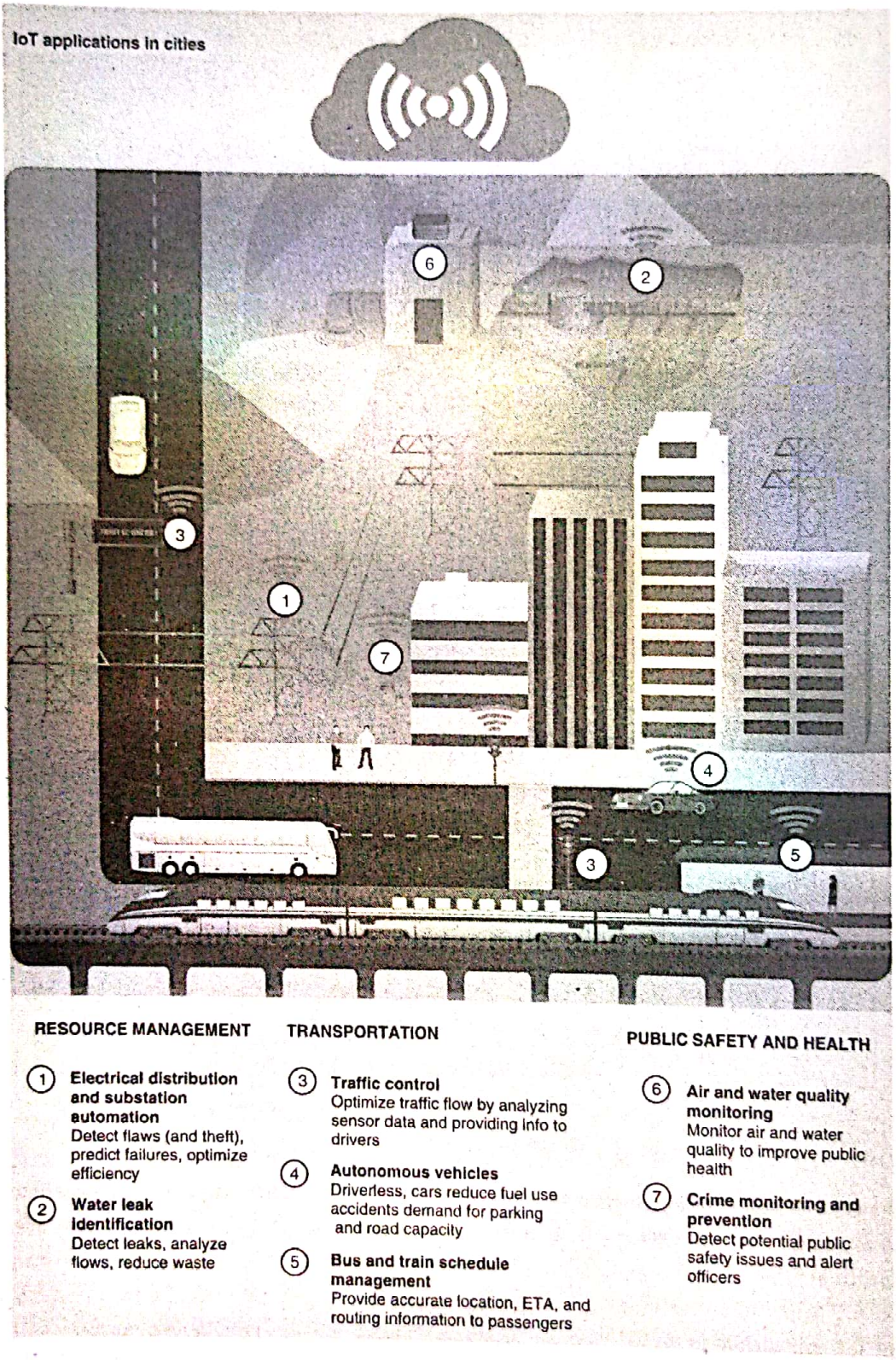
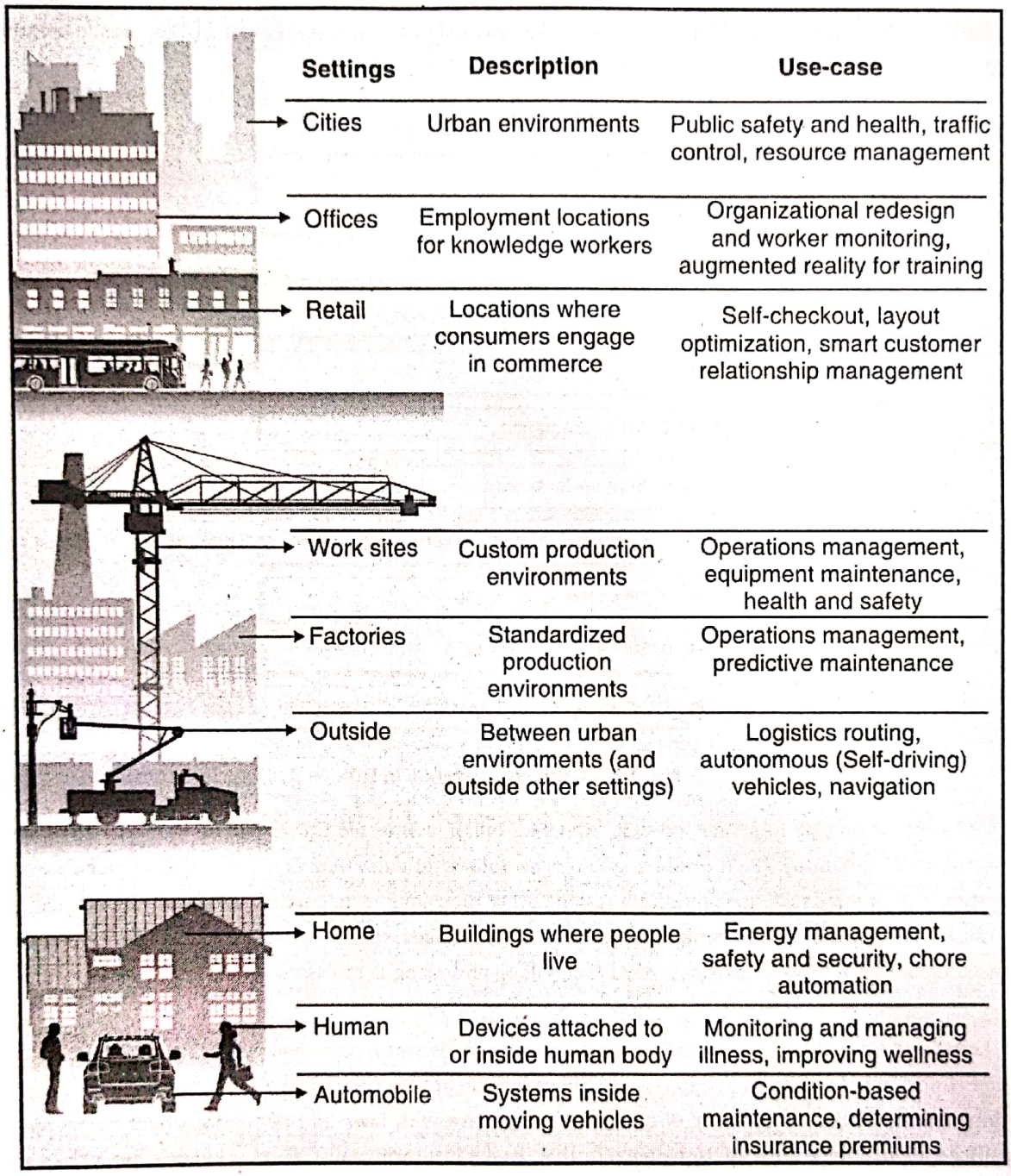



**Fig. 1.2.5 : IoT applications in cities**

To summarise, IoT applications could provide significant value in the areas which are shown in Fig. 1.2.6.

**Internet of Things applications create value in nine settings**



| Settings | Description | Use-case |
|---|---|---|
| Cities | Urban environments | Public safety and health, traffic control, resource management |
| Offices | Employment locations for knowledge workers | Organizational redesign and worker monitoring, augmented reality for training |
| Retail | Locations where consumers engage in commerce | Self-checkout, layout optimization, smart customer relationship management |
| Work sites | Custom production environments | Operations management, equipment maintenance, health and safety |
| Factories | Standardized production environments | Operations management, predictive maintenance |
| Outside | Between urban environments (and outside other settings) | Logistics routing, autonomous (Self-driving) vehicles, navigation |
| Home | Buildings where people live | Energy management, safety and security, chore automation |
| Human | Devices attached to or inside human body | Monitoring and managing illness, improving wellness |
| Automobile | Systems inside moving vehicles | Condition-based maintenance, determining insurance premiums |

**Fig. 1.2.6 : Major IoT Application Areas**

## 1.3 Emerging Trends in IoT

Gartner, a reputed research, and advisory company has predicted some of the major emerging trends in IoT in its report published at https://www.gartner.com/en/newsroom/press-releases/2018-11-07-gartner-identifies-top-10-strategic-IoT-technologies-and-trends. These trends are as shown in Fig. 1.3.1.

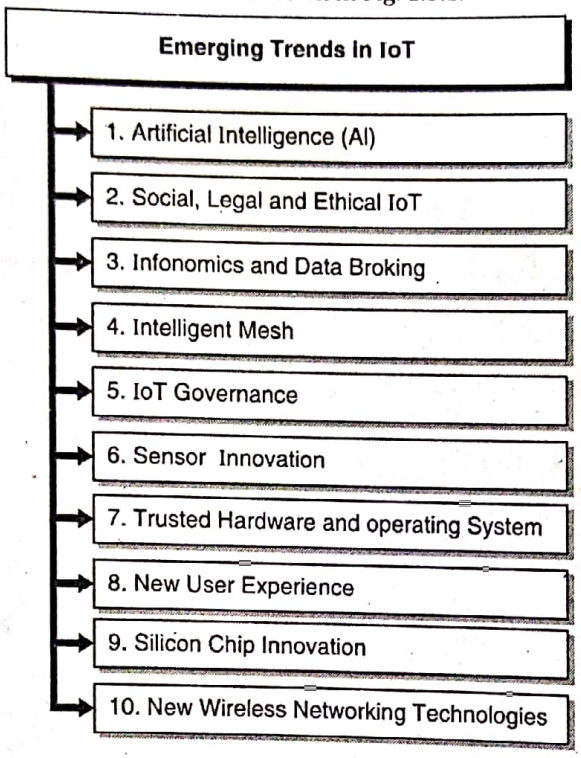
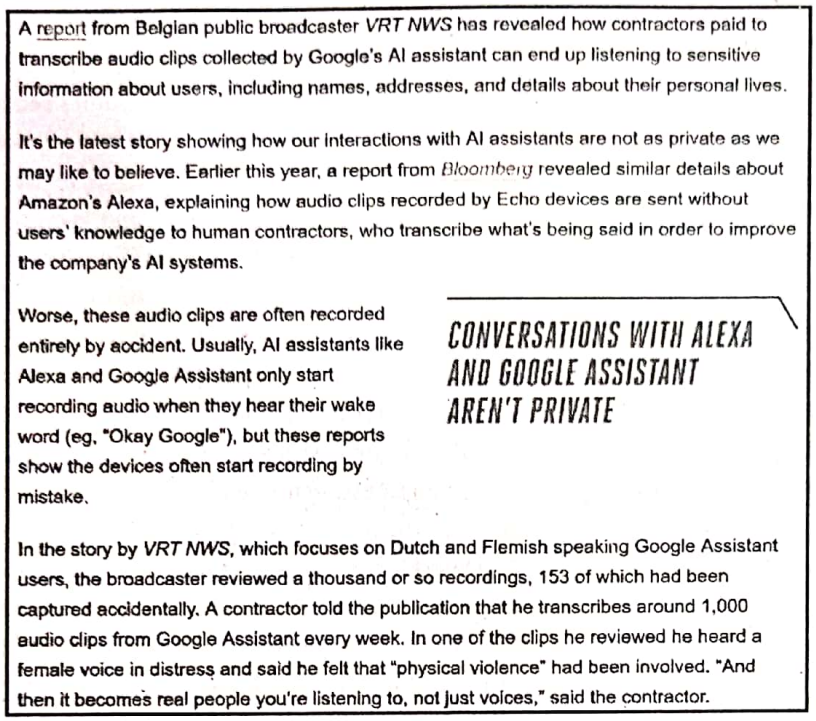



**Fig. 1.3.1 : Emerging trends in IoT**

1. **Artificial Intelligence (AI) :** Gartner forecasts that 14.2 billion connected things will be in use in 2019, and that the total will reach 25 billion by 2021, producing immense volume of data. Data is the fuel that powers the IoT and the organisation's ability to derive meaning from it will define their long term success. Artificial Intelligence (AI) will be applied to a wide range of IoT information, including video, images, speech, network traffic activity and sensor data. Various companies are heavily investing in AI technology and using it to deliver quicker and higher value from IoT data.

2. **Social, Legal and Ethical IoT :** As the IoT market matures and becomes more widely deployed, a wide range of social, legal, and ethical issues will grow in importance. These include ownership of data and the conclusions derived from it, algorithmic bias, security and privacy of data, and compliance with laws and regulations both locally and globally. Successful deployment of an IoT solution demands that it's not just technically effective but also socially acceptable.

   The key algorithms and AI systems could mandatorily be required to be reviewed by external consultancies to identify potential biases. You may have heard the news on Google Home and Amazon Alexa snooping on people's private lives where human contractors could actually listen to private recordings even when these devices were not activated. Scary, isn't it?



A report from Belgian public broadcaster *VRT NWS* has revealed how contractors paid to transcribe audio clips collected by Google's AI assistant can end up listening to sensitive information about users, including names, addresses, and details about their personal lives.

It's the latest story showing how our interactions with AI assistants are not as private as we may like to believe. Earlier this year, a report from *Bloomberg* revealed similar details about Amazon's Alexa, explaining how audio clips recorded by Echo devices are sent without users' knowledge to human contractors, who transcribe what's being said in order to improve the company's AI systems.

Worse, these audio clips are often recorded entirely by accident. Usually, AI assistants like Alexa and Google Assistant only start recording audio when they hear their wake word (eg, "Okay Google"), but these reports show the devices often start recording by mistake.

CONVERSATIONS WITH ALEXA AND GOOGLE ASSISTANT AREN'T PRIVATE

In the story by *VRT NWS*, which focuses on Dutch and Flemish speaking Google Assistant users, the broadcaster reviewed a thousand or so recordings, 153 of which had been captured accidentally. A contractor told the publication that he transcribes around 1,000 audio clips from Google Assistant every week. In one of the clips he reviewed he heard a female voice in distress and said he felt that "physical violence" had been involved. "And then it becomes real people you're listening to, not just voices," said the contractor.

**Fig. 1.3.2**

3. **Infonomics and Data Broking :** Lot of companies buy and sell real-life IoT data for either developing new products or for improving existing ones. The monetisation of this data could be seen as a strategic business asset. By 2023, the buying and selling of IoT data will become an essential part of many IoT systems. Organisations must understand the risks and opportunities related to data broking in order to set the IT policies required in this area and to advise other parts of the organisation.
4. **Intelligent Mesh :** The IoT infrastructure based computing is already shifting from centralised and cloud computing to edge computing. However, a more unstructured architecture, comprising of a wide range of "things" and services connected, requires a dynamic mesh kind of computing architecture. These mesh architectures will enable more flexible, intelligent, and responsive IoT systems.
5. **IoT Governance :** As the IoT deployment continues to expand, the need for a governance framework that ensures appropriate behaviour in the creation, storage, use and deletion of information related to IoT projects will become increasingly important. Governance ranges from simple technical tasks such as device audits and firmware updates to more complex issues such as the control of devices and the usage of the information they generate.

   Very similar to how IT teams are managing servers, desktops, networking, or sensitive data within the organisation according to the organisation's policies, these teams would soon need to ensure adequate governance for IoT devices and the data they generate.
6. **Sensor Innovation :** The sensor market is evolving continuously. New sensors will enable a wider range of situations and events to be detected, current sensors will fall in price to become more affordable or will be packaged in new ways to support new applications, and new algorithms will emerge to deduce more information from current sensor technologies.
7. **Trusted Hardware and Operating System :** Security is one of the most significant area of technical concern for organisations deploying IoT systems. This is because organisations often do not have enough control over the source and nature of the software and hardware being utilised in IoT deployments.

However, it is expected that organisations would have certification and attestation processes in place to ensure that the deployment of hardware and software combinations create more trustworthy and secure IoT systems. Organisations would ensure that the right staffs are involved in reviewing any decisions that involve purchasing IoT devices and embedded operating systems and ensuring that they meet the organisation's security policies.

8. **New User Experiences :** As the IoT markets evolve, the IoT user experience will be driven by four factors - new sensors, new algorithms, new experience architectures and context, and socially aware experiences. With an increasing number of interactions occurring with things that do not have regular screens and keyboards, organizations will be required to use new technologies and adopt new perspectives to build superior experiences to reduces friction and cognitive overload for the user and also encourage usage, adoption and user retention.
9. **Silicon Chip Innovation :** Currently, most IoT endpoint devices use conventional processor chips, with low-power ARM architectures. However, traditional instruction sets, and memory architectures are not well-suited to run all the tasks that IoT endpoints need to perform. New special-purpose chips will further reduce the power consumption and will support new capabilities such as data analytics integrated with sensors, and speech recognition included in low cost battery-powered devices. Silicon chips enabling functions, such as embedded AI, will in turn enable organisations to create highly innovative products and services.
10. **New Wireless Networking Technologies :** IoT networking involves balancing a set of competing requirements, such as endpoint cost, power consumption, bandwidth, latency, connection density, operating cost, quality of service, and range. No single networking technology optimises all of these today. However, the new IoT networking technologies, such as 5G, the forthcoming generation of low earth orbit satellites, and backscatter networks, may provide additional choice and flexibility..

## 1.4 Economic Significance of IoT

As per the McKinsey report stated earlier, The IoT market has the potential to generate about $4 trillion to $11 trillion in economic value by 2025. The Fig. 1.4.1 shows the various areas of potential for IoT with predictions for low and high impact.

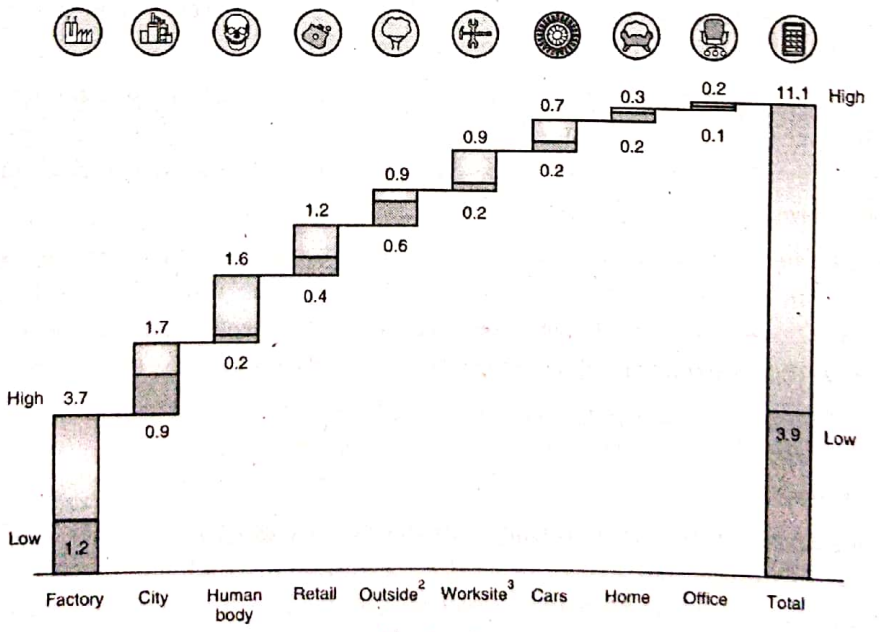



**Fig. 1.4.1**

Additionally, it is expected that there would be newer areas of IoT applications having multi-trillion opportunities.

New ecosystems are likely to emerge in place of many traditional industries by 2025

Ecosystem illustration, estimated total sales in 2025,[1] $ trillions

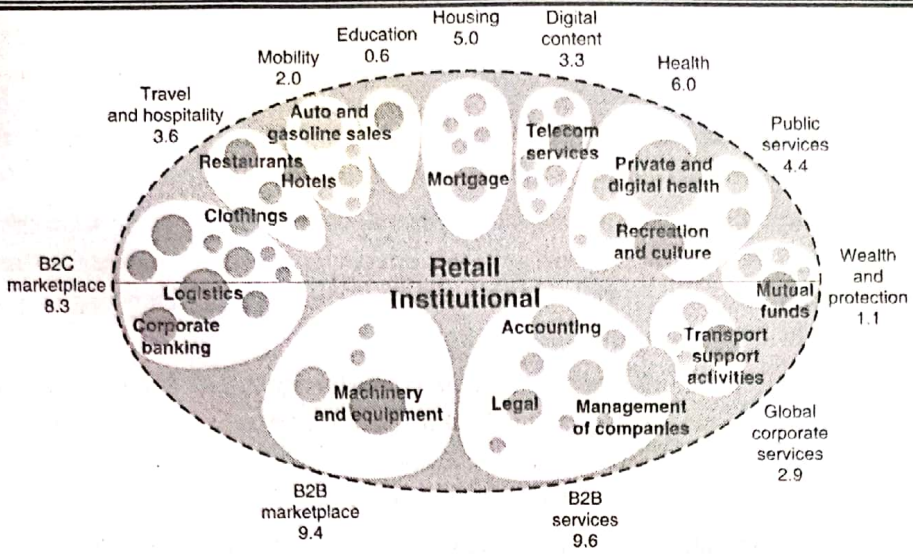



**Fig. 1.4.2**

Circle sizes show approximate revenue pool sizes, additional ecosystems are expected to emerge in addition to the those depicted; not all industries or subcategories are shown.

Source : HIS World Industry Service; Panorama by McKinsey; McKinsey analysis.

The companies can monetise different aspects of their IoT products and services. For example, assume that a company comes up with an Internet connected backpack for school children. So, the company could likely do the following.

**Deepen customer understanding**

**How?**
- How often is the backpack set down and picked up?
- What movement patterns can be identified (running, walking, sitting)?

**When?**
- Is it worn only on school days or also during the holidays?
- How is it used during the day?
- How long does it take to get to school?

**Generate additional sales**

**Features available at an additional charge**
- Alarm feature when child veers off the path to school—one-time charge for use

**Subscription model**
- Automatic opening of the front door can be booked separately and monthly after the sale

**Boost customer loyalty**

**Engaging**
- Maintain regular contact with customers via app

**Exploring**
- Determine and analyze customer preferences

**Retaining**
- Offer new product features via software updates

## 1.5 Internet of Things Definitions and Frameworks

IoT is still evolving and there are several definitions and frameworks available that address one or more aspects of IoT. Let's look at some of the definitions and common frameworks. You would read about various frameworks in detail in subsequent chapters. This section would provide a basic overview.

### IoT Definitions

The MIT AUTO-ID LABORATORY coined the term Internet of Things (IoT) and traces its roots back to 1999 with the founding of the Auto-ID Center, which laid much of the groundwork for the standardisation of RFID technology and the introduction of the EPC (Electronic Product Code). Since, then IoT has seen several advancements.

**General Observations**

Some of the general advancements and observations with respect to IoT are as following :

1. Internet of Things is a twenty-first century phenomenon in which physical consumer products connect to the web and start communicating with each other by means of sensors and actuators.
2. The term "Internet of Things" denotes a trend where a large number of devices benefit from communication services that use Internet protocols. Many of these devices are not directly operated by humans, but exist as components in buildings, vehicles, and the environment. There will be a lot of variation in the computing power, available memory, and communications bandwidth between different types of devices.
3. The term M2M is used to refer to machine-to-machine communication, i.e., automated data exchange between machines. The term "machine" may also refer to virtual machines such as software applications. Viewed from the perspective of its functions and potential uses, M2M is causing an entire "Internet of Things", or internet of intelligent objects, to emerge.
4. IoT spans a great range of applications. People bring varied assumptions about what devices are 'things'. Most IoT devices have constraints but the nature of constraints varies. IoT needs to be divided into manageable topic areas.
5. The vision of the internet of things is to attach tiny devices to every single object to make it identifiable by its own unique IP address. These devices can then autonomously communicate with one another. The success of the internet of things relies on overcoming the following technical challenges.
   (a) There should be enough IP addresses available so as to provide an IP address to every possible object that may need it in the future.
   (b) The power requirements for embedded chips on such devices will need to be smaller and very efficient.
   (c) The software applications must be developed that can communicate with and manage the stream of data from hundreds of interconnected non-computing devices that comprise a 'smart' system which can adapt and respond to changes.
6. IoT is going to be an advanced network including normal physical objects together with computers and other advanced electronic appliances. Instead of forming ad hoc network, normal objects will be a part of whole network so that they can collaborate, understand real time environmental data and react accordingly as needed.

## 1.5.1 ITU-T Views

The Study Groups of ITU's Telecommunication Standardization Sector (ITU-T) assemble experts from around the world to develop international standards known as ITU-T Recommendations which act as defining elements in the global infrastructure of information and communication technologies (ICTs). Standards are critical to the interoperability of ICTs and whether we exchange voice, video or data messages, standards enable global communications by ensuring that countries' ICT networks and devices are speaking the same language.

ITU-T has two views (perspectives) for defining IoT.

- **View A :** IoT is just a concept. The IoT does not refer to a network infrastructure and neither the IoT is a technical term.
- **View B :** IoT is an infrastructure: The IoT refers to an infrastructure.

ITU-T's various study groups have attempted to define IoT based on these views. The Table 1.2.1 provides some of the examples of IoT definitions based on these two views.

**Table 1.5.1**

| View Type | IoT Definitions (examples) |
|---|---|
| View A (IoT as a concept) | A technological revolution that represents the future of computing and communications, and its development depends on dynamic technical innovation in a number of important fields, from wireless sensors to nanotechnology.<br>The networked interconnection of objects—from the sophisticated to the mundane—through identifiers such as sensors, RFID tags, and IP addresses. The Internet of things links the objects of the real world with the virtual world, thus enabling anytime, anyplace connectivity for anything and not only for anyone. It refers to a world where physical objects and beings, as well as virtual data and environments, all interact with each other in the same space and time. The IoT refers to as ubiquitous networking or pervasive computing environments, is a vision where all manufactured things can be network enabled, that is connected to each other via wireless or wired communication networks. The IoT is a world where physical objects are seamlessly integrated into the information network, and where the physical objects can become active participants in business processes. |
| View B (IoT as an infrastructure) | A global network infrastructure, linking physical and virtual objects through the exploitation of data capture and communication capabilities. A global information and communication infrastructure enabling automated chains of actions (not requiring explicit human intervention) facilitating information assembly and knowledge production and contributing to enrichment of human life by interconnecting physical and logical objects based on standard and interoperable communication protocols and through the exploitation of data capture and communication capabilities supported by existing and evolving information and communication technologies. A global ICT infrastructure linking physical objects and virtual objects (as the informational counterparts of physical objects) through the exploitation of sensor and actuator data capture, processing and transmission capabilities. The Internet of Things consists of networks of sensors attached to objects and communication devices, providing data that can be analysed and used to initiate automated actions. The data also generate vital intelligence for planning, management, policy, and decision-making. |

**Working Definition**

So far you saw various ways in which the term IoT could be possibly defined. For our discussion in this book, I will use the following working definition of IoT.

✍ ***Definition :*** *The Internet of Things (IoT) is a collection of diverse technologies that interact with the physical world.*

This definition helps us to understand the various technologies that make IoT possible. These technologies could be around physical aspects of IoT, such as sensors and actuators, as well as around infrastructural elements of IoT such as network and communication protocols. You will read about several of these aspects throughout the book.

## 1.6 IoT Frameworks

Historically, quite a few Standard Development Organisations (SDOs) have attempted to standardise various aspects of IoT. However, there is no single body which is responsible for making IoT standards. There are considerable efforts at national and international level, at government level, and at different organisational levels for IoT standardization. Alliances have been formed by many domestic and multinational companies to agree on common standards and technology for the IoT. However, no universal body has been formed yet.

While organisations such as IEEE, Internet Engineering Task Force (IETF), ITU-T, OneM2M, 3GPP, etc., are active at the international level, Telecommunication Standards Development Society, India (TSDSI), Global ICT Standardization Forum for India (GISFI), Bureau of Indian Standards (BIS), Korean Agency for Technology and Standards (KATS), and so on, are active at national level. European Telecommunications Standards Institute (ETSI) has also come up with several standards at the regional level. Let's learn about some of the standards that these SDOs have defined so far.

**Table 1.6.1 : IoT protocol standardisation efforts by various SDOs**

| Organisation Name | Major IoT Standards Published |
|---|---|
| **International Telecommunication Union (ITU)** | It has released several standards under various study groups (SG). The most prominent study groups with respect to IoT are as following.<br>TSAG – Telecommunication Standardisation Advisory Group<br>SG2 – Operational aspects of service provision and telecommunications management<br>SG3 – Tariff and accounting principles including related telecommunication economic and policy issues<br>SG5 – Protection against electromagnetic environment effects<br>SG9 – Television and sound transmission and integrated broadband cable networks<br>SG11 – Signalling requirements, protocols and test specifications<br>SG12 – Performance and quality of service<br>SG13 – Future networks including mobile and NGN<br>SG15 – Optical and other transport network infrastructures<br>SG16 – Multimedia coding, systems and applications<br>SG17 – Security<br>SG20 – IoT and its applications including smart cities and communities |
| **Institute of Electrical and Electronics Engineers (IEEE)** | IEEE has been producing standards for local and personal area connectivity while playing a key role in forming the physical and Medium Access Control (MAC) layer standards (802.15.4, 802.3, etc.). However, it has also published several standards for IoT. Some of them are as following.<br>IEEE 11073-10103™-2012 - IEEE Standard for Health informatics<br>IEEE 1451.0™-2007 - IEEE Standard for a Smart Transducer Interface for Sensors and Actuators<br>IEEE 1547.4™-2011 - IEEE Guide for Design, Operation, and Integration of Distributed Resource Island Systems with Electric Power Systems<br>IEEE 1609.2™-2013 - IEEE Standard for Wireless Access in Vehicular Environments<br>IEEE 1703™-2012 - IEEE Standard for Local Area Network/Wide Area Network (LAN/WAN) |

| | |
|---|---|
| | Node Communication Protocol to complement the Utility Industry End Device Data Tables<br>IEEE 1905.1™-2013 - IEEE Draft Standard for a Convergent Digital Home Network for Heterogeneous Technologies<br>IEEE 21450™-2010 - IEEE Standard for Information technology - Smart transducer interface for sensors and actuators -- Common functions, communication protocols, and Transducer Electronic Data Sheet (TEDS) formats<br>IEEE 21451-1™-2010 - IEEE Standard for Information technology - Smart transducer interface for sensors and actuators - Part 1: Network Capable Application Processor (NCAP) information model<br>IEEE 21451-2™-2010 - IEEE Standard for Information technology - Smart transducer interface for sensors and actuators -- Part 2: Transducer to microprocessor communication protocols and Transducer Electronic Data Sheet (TEDS) formats<br>IEEE 21451-4™-2010 - IEEE Standard for Information technology -- Smart transducer interface for sensors and actuators -- Part 4: Mixed-mode communication protocols and Transducer Electronic Data Sheet (TEDS) formats<br>IEEE 21451-7™-2011 - IEEE Standard for Smart Transducer Interface for Sensors and Actuators--Transducers to Radio Frequency Identification (RFID) Systems Communication Protocols and Transducer Electronic Data Sheet Formats<br>IEEE 802.15.4e™-2012 - IEEE Standard for Local and metropolitan area networks--Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer<br>IEEE 802.15.4f™-2012 - IEEE Standard for Local and metropolitan area networks-- Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 2: Active Radio Frequency Identification (RFID) System Physical Layer (PHY)<br>IEEE 802.15.4g™-2012 - IEEE Standard for Local and metropolitan area networks--Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 3: Physical Layer (PHY) Specifications for Low-Data-Rate, Wireless, Smart Metering Utility Networks<br>IEEE 802.15.4™-2011 - IEEE Standard for Local and metropolitan area networks--Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)<br>IEEE 802.3™-2012 - IEEE Standard for Ethernet |
| **3rd Generation Partnership Project (3GPP)** | 3GPP unites seven telecommunications standard development organizations (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC), known as "Organizational Partners" and provides their members with a stable environment to produce the Reports and Specifications that define 3GPP technologies. With respect to IoT, it provides guidance on the following.<br>NarrowBand IoT (NB-IoT) - It is a radio standard developed for Low-Power Wide Area Network (LPWAN) to support IoT technologies. NB-IoT is designed for indoor coverage using large number of connected devices with low cost and long battery life.<br>enhanced Machine Type Communication (eMTC) – It is a low power wide area technology which supports IoT through lower device complexity and provides extended coverage, leveraging a mobile carriers existing LTE base stations.<br>Extended Coverage – GSM – Internet of Things (EC-GSM-IoT ) – It is based on eGPRS and designed as a high capacity, long range, low energy and low complexity cellular system for IoT communications. |

| | |
|---|---|
| **Internet Engineering Task Force (IETF)** | IETF is a complementing organization to IEEE, 3GPP, and ITU by creating the enabling protocols that actually connect the constrained nodes in a constrained environment in an efficient manner on top of the available physical and data-link layer technologies available from other SDOs working in that area. Also, IETF has been providing several standards for secure communication that are relevant for IoT domain. IETF has several standards but some of the most referred ones with respect to IoT are as following.<br>RFC791 - IPv4<br>RFC8200 - IPv6<br>RFC8576 - Internet of Things (IoT) Security<br>RFC 2616 - HTTP<br>RFC 2818 - HTTPS<br>RFC 7272 - CoAP<br>RFC 8323 - CoAP over TLS<br>RFC 6455 - WebSocket<br>RFC 6120 - XMPP<br>RFC 5246 - TLS<br>RFC 6568 - Design and Application Spaces for 6LoWPANs<br>RFC 7925 - TLS / DTLS Profiles for IoT |
| **Organization for the Advancement of Structured Information Standard (OASIS)** | OASIS has two standards that are commonly referred for IoT domain – MQTT and AMQP. |
| **oneM2M** | oneM2M provides specifications for architecture, APIs, security, and interoperability guidelines and certification for M2M/IoT devices and applications. The purpose and goal of oneM2M is to develop technical specifications which address the need for a common M2M Service Layer that can be readily embedded within various hardware and software and relied upon to connect the various devices in the field with M2M application servers worldwide. Eight SDOs (ARIB, ATIS, CCSA, ETSI, TIA, TSDSI, TTA, TTC) have come together for collaborating under oneM2M mission. Some of the released drafts are as following.<br>oneM2M-SensorThings API interworking<br>Effective IoT Communication to Protect 3GPP Networks<br>System enhancements to support Data License Management<br>System enhancements to support Data Protection Regulations |

| | |
|---|---|
| **European Telecommunications Standards Institute (ETSI)** | European Telecommunications Standards Institute (ETSI) is an independent, not-for-profit, standardization organization in the telecommunications industry in Europe. It is part of several alliances working to develop standards for IoT. It is focusing on variods aspects and usage of IoT such as the following.<br>Smart appliances<br>Smart M2M<br>Smart metering<br>Smart cities<br>Smart grids<br>eHealth<br>Intelligent Transport Systems<br>Wireless Industrial Automation<br>Security for the IoT<br>Ultra-Low Energy Digital Cordless Telecommunications |
| **International Organization for Standardization (ISO)** | The International Organization for Standardization is an international standard-setting body composed of representatives from various national standards organizations. It has established several standards in various domains. With respect to IoT, some of the standards that it has published are as following.<br>ISO/IEC 30141, Internet of Things (IoT) – Reference architecture, provides an internationally standardised IoT Reference Architecture using a common vocabulary, reusable designs and industry best practice.<br>ISO/IEC 21823 – Interoperability for IoT systems<br>ISO/IEC 27030 – Guidelines for security and privacy in IoT<br>ISO/IEC 30166 – Industrial IoT<br>ISO/IEC 30165 – Real-time IoT framework<br>ISO/IEC 30148 – Technical requirements and application of sensor network for wireless gas meters<br>ISO/IEC 29182 – Characteristics and requirements of a sensor networks |

## 1.7 IoT Architecture

The Fig. 1.7.1 shows block diagram provides a high-level architectural view of IoT.

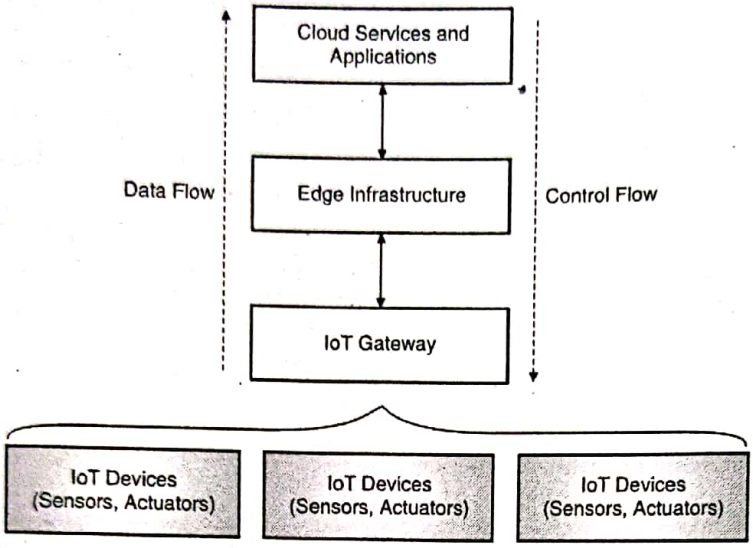



**Fig. 1.7.1 : High-level architecture of IoT**

IoT architecture consists of four major components as shown in the architecture Fig. 1.7.1.

1. **IoT Devices :** These are the IoT devices such as thermostats, cameras, watch, refrigerator, cars, etc. These devices continuously record and track various data as per their design. For example, your smart watch continuously tracks your pulse. These devices have various connectivity options such as ZigBee, Bluetooth, and Wi-Fi.
   IoT devices typically have
   (a) Sensors : These are electronic circuits that continuously read the data as designed. These could be RFID, GPS, Accelerometers, Gyroscope, etc.
   (b) Actuators : These are electronic circuits that can not only read data but also take actions. For example, if the room temperature goes beyond a certain limit, an actuator can automatically power on or power off air conditioning system.

   You have already learnt about how sensors and actuators work in detail in the "Architecture of a Real-time System" section previously.

2. **IoT Gateway :** There could be several IoT devices in an industrial setting. For example, if you are using a connected car, there could be several cars on the road from the same manufacturer. The various data points from the connected cars such as engine health, mileage, driving style and other car performance related parameters could be tracked by the manufacturer for providing preventive maintenance and breakdown support. To optimise the number of direct connections to the devices, an IoT Gateway is used. An IoT gateway aggregates data from several devices and processes them. The pre-processing is required so that only the meaningful information is sent further. IoT devices could collect data at a milli-second level. Such high granularity of data may not be required to be sent further in its entirety. IoT gateways, thus, process the granular data from several devices and send them further.
3. **Edge Infrastructure :** Once the IoT data is synthesised at the IoT gateway, it is sent to edge infrastructure. Edge infrastructure provides cloud computing resources closer to the user to avoid sending large volume of data over the network. You can run analysis functions, execute predictions based on machine learning models, keep device data in sync, and communicate with other devices securely. You can further filter device data and only send necessary information to the cloud.

4. **Cloud Services and Applications :** Finally, the IoT data can further be analysed using cloud services and resources. You could have several IoT based applications such as car predictive maintenance, health monitoring, traffic monitoring system, home automation system or anything else that could consume the data from the IoT devices and produce meaningful information and actions.

5. **Network Connectivity :** One thing that is transparent in the architecture diagram is the network connectivity from IoT devices all the way up to the cloud services and applications. Without network connectivity, there is no internet in "Internet of Things". There could be several networking protocols at various layers providing seamless network connectivity to operate, control and manage the IoT devices and make optimum use of them.

Some of the common connectivity options are 2G, 3G, 4G, LTE, 5G, Wi-Fi, Bluetooth, NFC, ZigBee, USB, Ethernet, etc. These devices also support a wide range of network communication protocols such as HTTP, CoAP, MQTT, AMQP, etc. The Fig. 1.7.2 summarises the building blocks (or the high-level architecture) of IoT.

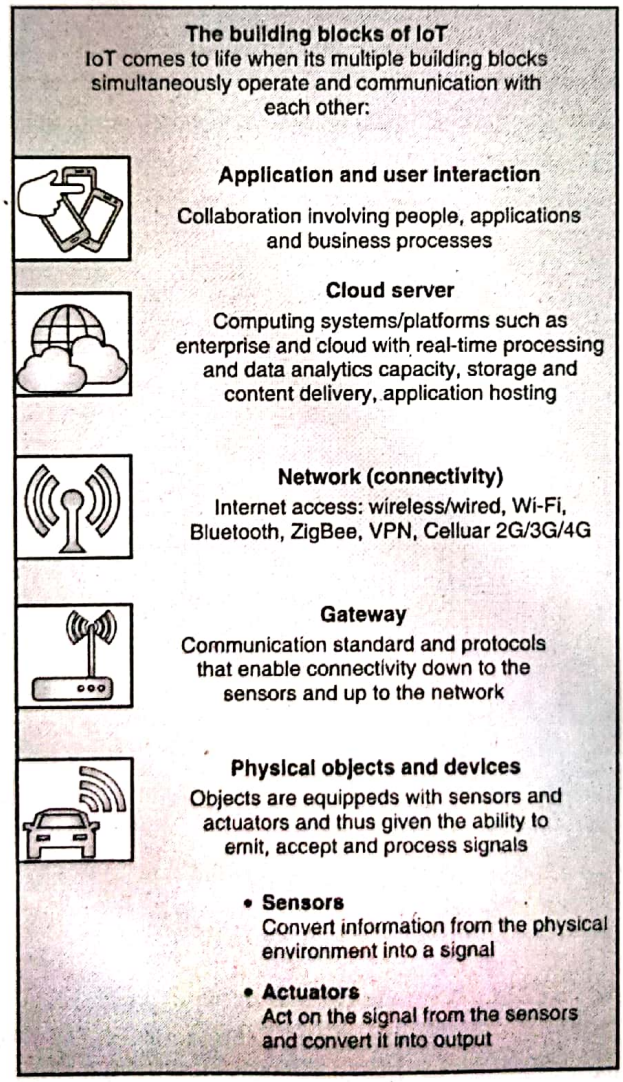



**Fig. 1.7.2**

## 1.8 Physical Design of IoT

Now that you have a pretty good hang of what IoT is and what kind of purposes it can be used for, let's dive a little deeper in this section to understand how these devices communicate with the external world. Each technology or protocol described in this section could possibly be a book of its own if not multiple volumes of books. But, I will walk you through them at a detail that you need to know to build a general concept around them. Let's go.

### 1.8.1 Basic Nodal Capabilities

As you understand, things in IoT could refer to the actual device or equipment that performs a specific sensing or actuating function. IoT devices are an outcome of combining the worlds of information technology (IT) and operational technology (OT).

Many IoT devices are the result of the convergence of cloud computing, mobile computing, embedded systems, big data, low-price hardware, and other technological advances. IoT devices can provide computing functionality, data storage, and network connectivity for equipment that previously lacked them, enabling new efficiencies and technological capabilities for the equipment, such as remote access for monitoring, configuration, and troubleshooting. IoT can also add the abilities to analyse data about the physical world and use the results to better inform decision making, alter the physical environment, and anticipate future events.

At a high-level, there are three types of capabilities in IoT devices. The Fig. 1.8.1 lays out these capabilities.

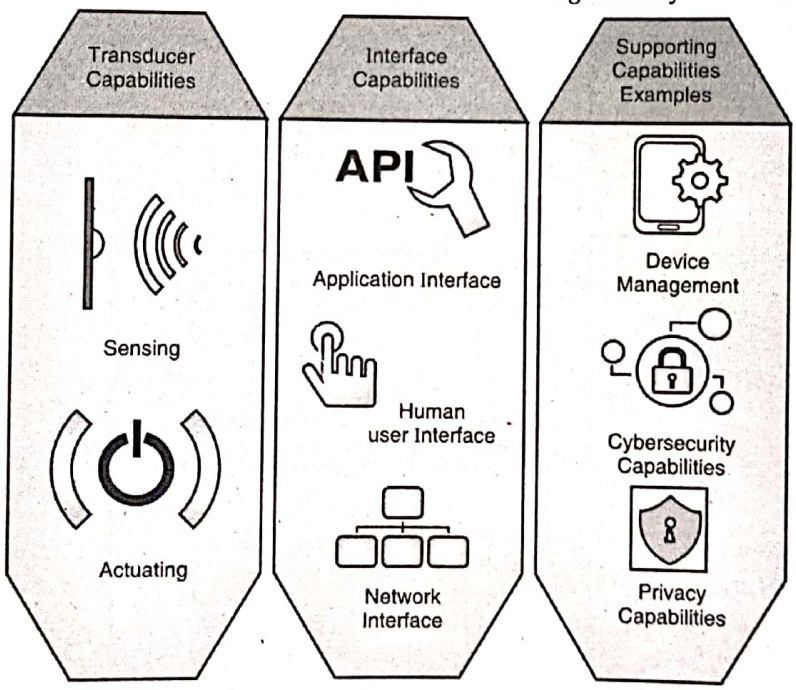



**Fig. 1.8.1**

1. **Transducer Capabilities :** Transducer capabilities interact with the physical world and serve as the bridge between digital and physical environments. Transducer capabilities provide the ability for computing devices to interact directly with physical entities of interest. Every IoT device has at least one transducer capability. The two types of transducer capabilities are

   (a) Sensing capability : It is the ability to provide an observation of an aspect of the physical world in the form of measurement data. For example, temperature measurement, radiographic imaging, optical sensing, and audio sensing.

(b) Actuating capability : It is the ability to change something in the physical world. For example, powering on a bulb, turning off heating, automatically opening doors as people approach and robotics.

2. **Interface capabilities :** Interface capabilities enable interactions (communication) with IoT devices. These could be device-to-device communications, external world to device communication (or vice-versa), or human-to-device communications. The major types of interface capabilities are :

(a) Application interface : Application interface is the ability for other computing devices to communicate with an IoT device through an IoT device application.

An example of an application interface capability is an application programming interface (APIs). APIs are used extensively to instrument IoT devices for both – receiving output as well as providing input.

(b) Human user interface : Human user interface is the ability for an IoT device and people to communicate directly with each other. For example, touch screens, microphones, cameras, and speakers. This is precisely how you direct the IoT device to carry out an action such as by saying "Alexa, what does my day look like" and receiving the response through speaker.

(c) Network interface : Network interface is the ability to interface with a communication network for the purpose of communicating data to or from an IoT device. This is what connects the IoT device to a network and makes it "Internet of" and useful. A network interface capability includes both hardware and software. For example, a network interface card or a chip and the software implementation of the networking protocol that uses the card or chip. Examples of network interface capabilities include Ethernet, Wi-Fi, Bluetooth, Long-Term Evolution (LTE), and ZigBee. Every IoT device has at least one enabled network interface capability (else how will it be "Internet of", isn't it) and may have more than one.

3. **Supporting capabilities :** Supporting capabilities provide functionality that supports the other IoT device related capabilities. Some of these capabilities could be related to device management (deployment, configure, control, patch, update, etc.), security and privacy. These capabilities work together to provide a seamless user experience.

## 1.9 IoT Protocols

Do you recall OSI network model ? Before you understand IoT protocols, let's quickly refresh OSI network model so that it makes more sense to you when I discuss IoT protocols.

**Note :** Discussing OSI Model in-depth is beyond the scope of this book. It is assumed that you have covered it in detail in your subjects on networking. A general high-level overview is presented here as a refresher.

✍ ***Definition :*** *The Open Systems Interconnection model (OSI model) is a conceptual model that characterises and standardises the communication functions of networked communications without diving into complexities of protocols, architecture, and the underlying technologies.*

The OSI model consists of 7 layers. Each layer interacts with the layer above and below it and passes on the respective protocol data units encapsulated into their respective headers.

**Table 1.9.1**

| Layer Number | Layer Name | Protocol Data Unit | Function |
|---|---|---|---|
| 7 | Application | Data | Application Interface – APIs, UIs |
| 6 | Presentation | Data | Data translation between networking and application |
| 5 | Session | Data | Manage communication sessions between sender and receiver |

| Layer Number | Layer Name | Protocol Data Unit | Function |
|---|---|---|---|
| 4 | Transport | Segment, Datagram | Reliable data transmission |
| 3 | Network | Packet | Network packet addressing and routing |
| 2 | Data Link | Frame | Transmission of data between two nodes |
| 1 | Physical | Binary | Actual communication over physical media |

Note that each OSI layer protocol adds its own information to the data packet.

***Definition :*** *The process of adding layer specific information and passing on the data to the next layer below is called encapsulation.*

Encapsulation happens top – down (From Application to Physical Layer).

***Definition :*** *The process of removing layer specific information and passing on the data to the next layer above is called decapsulation.*

Decapsulation happens bottom – up (from Physical to Application Layer).

Table 1.9.2 shows a quick reference summary for various protocols at the respective OSI Layers.

**Table 1.9.2**

| Layer Name | Protocols Used |
|---|---|
| Application | HTTP, FTP, SMTP, etc. |
| Presentation | JPEG, MPEG, TIFF, ASCII, etc. |
| Session | NFS, RPC, etc. |
| Transport | TCP, UDP, SSL, etc. |
| Network | IP, ICMP, OSPF, etc. |
| Data Link | ARP, PPP, Ethernet, etc. |
| Physical | ISDN, DSL, 10Base-T, etc. |

Alright, let's come back to IoT protocols. IoT devices, because of their small form factor and limited processing capabilities, use only certain protocols for communication. Not all general protocols used in regular computing are used for IoT devices. Some of the major protocols used for IoT are as shown in Table 1.9.3. They are broken up as per the OSI layer for easily understanding their roles. Note here that not all the possible protocols are listed here. These are the major and most commonly used ones.

**Table 1.9.3**

| |
|---|
| **Application Layer**<br>HTTP, CoAP, WebSockets, MQTT, XMPP, DDS, AMQP |
| **Transport Layer**<br>TCP, UDP |
| **Network Layer**<br>IPv4, IPv6, 6LoWPAN |
| **Link Layer**<br>802.3 – Ethernet, 802.11 – WLAN, 802.15.4 – LR-WPAN, 802.16 – WiMAX, 2G / 3G / 4G / LTE / 5G - Cellular |

Let's learn about them briefly here.

### 1.9.1 Link Layer Protocols

At the Link Layer, the following protocols are commonly used for IoT devices.

1. **802.3 – Ethernet**

IEEE 802.3 refers to a family of specifications developed by IEEE for wired Ethernet technology. There are several specifications that come under its umbrella. Some of them are 802.3a, 802.3i, 802.3j, 802.3ae and 802.3z. Each of these standards have its own set of detailed specifications. Ethernet is a resource-sharing technology that enables several devices to communicate on the same network. Ethernet usually uses a bus or star topology.

If a linear bus topology is used, then all devices connect to one cable. If a star topology is used, then each device is connected to a cable that is connected to a centralised device, such as a switch. 802.3 standards define various mediums for connecting devices such as coaxial cables, twisted pairs, and optical fibre. The speed could range from 10 Mbps to over 10 Gbps. It is one of the most widely used technology for connecting devices to the network.

2. **802.11 – WLAN**

IEEE 802.11 refers to a family of specifications developed by IEEE for WLAN (Wireless LAN) technology. There are several specifications that come under its umbrella. Some of them are 802.11a, 802.11b, 802.11g, 802.11n and 802.11i. Each of these standards have its own set of detailed specifications. WLAN, usually implemented as Wi-Fi, has becomes the most preferred way to connect devices to the network because of its ease of use and roaming capabilities that does not require a physical wired connection. You can conveniently use it anywhere in your office, home, and public places such as airports and railway stations. Wireless networking can be provided through small wireless networking chips on the IoT devices. You do not require external interface for plugging network cables. This reduces the size of the IoT device.

3. **802.15.4 – LR-WPAN**

IEEE 802.15.4 refers to a family of specifications developed by IEEE for LR-WPAN (Low Rate Wireless Personal Area Network) technology. There are several specifications that come under its umbrella. Some of them are 802.15.4a, 802.15.4c, 802.15.4d, 802.15.4e, 802.15.4f and 802.15.4g. Each of these standards have its own set of detailed specifications. This protocol offers several benefits such as low power consumption, low data rate, low-cost and high-message throughput. It provides data rates of 250 kbps, 40 kbps, and 20 kbps.

This protocol also provides reliable communication and can handle a huge number of IoT devices (approximately about 65,000 devices). This protocol forms the basis of ZigBee and other protocols that are commonly used in IoT communication.

4. **802.16 – WiMAX**

IEEE 802.16 refers to a family of specifications developed by IEEE for broadband Wireless Metropolitan Area Networks technology. There are several specifications that come under its umbrella. Some of them are 802.16.1a, 802.16.1b, 802.16p, 802.16n, 802.16a and 802.16s. Each of these standards have its own set of detailed specifications. Although the 802.16 family of standards is officially called WirelessMAN in IEEE, it has been commercialised under the name "WiMAX" (from "Worldwide Interoperability for Microwave Access") by the WiMAX Forum industry alliance. The Forum promotes and certifies compatibility and interoperability of products based on the IEEE 802.16 standards. 802.16 standards provide data rates from 1.5 Mbps to 1 Gbps.

**5. 2G / 3G / 4G / LTE / 5G – Cellular**

Cellular connectivity via 2G / 3G / 4G / LTE / 5G could be used for connecting IoT devices as well. Unlike traditional telephone networks, cellular networks could be used for data connectivity as well as voice connectivity. Cellular networks have evolved over several decades. As of today, we have three network types which are predominantly in use - 2G, 3G and 4G. 5G is just around the corner.

**Table 1.9.4**

| Technology | Generation |
|---|---|
| Analog Circuit | 1G |
| GSM | 2G |
| UMTS | 3G |
| LTE | 4G |
| 5G New Radio | 5G |

Like wireless connectivity, cellular connectivity also provides ease of use and roaming capabilities that does not require a physical wired connection. You can conveniently use it anywhere in your office, home, and public places such as airports and railway stations. Cellular connectivity can be provided through small networking chips on the IoT devices. You do not require external interface for plugging network cables. This reduces the size of the IoT device.

## 1.9.2 Network Layer Protocols

At the Network Layer, the following protocols are commonly used for IoT devices.

**1. IPv4**

IP stands for Internet Protocol. IP defines a set of protocols that can be used for communication between any two devices on the network. IP provides addressing and routing mechanisms for each packet of data that needs to move across the network. Each device on the network must have a unique IP address to communicate with any other device on the network. IPv4 is IP version 4. This is the most common IP addressing scheme used today despite certain challenges. It is 32-bit long and thus has an address space of $2^{32} = 4,294,967,296$. This means you can maximally have 4,294,967,296 (approximately 4.3 billion) IPv4 addresses. However, there are many more devices than the number 4,294,967,296 today and hence IPv4 addressing is insufficient to address all the devices. An example of IPv4 address looks like 121.56.78.214.

**2. IPv6**

IP stands for Internet Protocol. IP defines a set of protocols that can be used for communication between any two devices on the network. IP provides addressing and routing mechanisms for each packet of data that needs to move across the network. Each device on the network must have a unique IP address to communicate with any other device on the network. IPv6 is IP version 6. IPv6 was created to address the limitation of IPv4 to have only 4,294,967,296 IP addresses due to 32-bit length. IPv6 more or less provides the similar addressing and routing capabilities but one core difference between IPv4 and IPv6 is the address space. IPv6 address is 128-bit long and thus you can have $2^{128}$ IPv6 addresses! That's enough number of IP addresses to cover entire spectrum of IoT devices. An example of IPv6 address looks like 2001:0:9d38:6abd:2c37:10da:8554:4234.

**3. 6LoWPAN**

6LoWPAN stands for IPv6 over Low-Power Wireless Personal Area Networks. It is an open standard defined in RFC 6282 by the Internet engineering task force (IETF). The key feature of 6LoWPAN that makes it suitable for IoT communication is that though it was originally designed to support IEEE 802.15.4 low-power wireless networks in the 2.4-GHz band, it now supports a wide range of networking media such as sub-1 GHz low-power RF, Bluetooth smart, power line control (PLC), and low-power Wi-Fi. 6LoWPAN can communicate with 802.15.4 devices as well as other types of devices on an IP networks such as over Wi-Fi.

### 1.9.3 Transport Layer Protocols

At the Transport Layer, the following protocols are commonly used for IoT devices.

**1. TCP**

TCP (Transmission Control Protocol) is a standard that defines how to establish and maintain a network connection through which application programs can exchange data. TCP is intended for use as a highly reliable host-to-host protocol between hosts in packet-switched computer communication networks, and in interconnected systems of such networks. TCP works with the Internet Protocol (IP), which defines how computers send packets of data to each other. Together, TCP and IP are the basic rules defining the Internet. The Internet Engineering Task Force (IETF) defines TCP in the Request for Comment (RFC) standards document number 793.

TCP is a reliable and connection-oriented protocol, which means that it ensures packets are delivered to the destination host. If a packet is lost during transmission, then TCP has the ability to identify this issue and resend the lost or corrupted packet. TCP also supports packet sequencing (to ensure each and every packet was received), flow and congestion control, and error detection and correction.

**2. UDP**

UDP (User Datagram Protocol) is a standard that provides a procedure for application programs to send messages to other programs with a minimum protocol mechanism overhead. The protocol is transaction-oriented (instead of connection-oriented), and delivery and duplicate protection are not guaranteed. The Internet Engineering Task Force (IETF) defines UDP in the Request for Comment (RFC) standards document number 768.

UDP is a best-effort and connectionless protocol. It has neither packet sequencing nor flow and congestion control, and the destination host does not acknowledge every packet it receives. It does not guarantee that a packet will reach its destination and thus is treated as an unreliable protocol. But, it uses fewer resources and is faster than TCP as it does not have communication overhead such as error check and packet sequencing. UDP is suitable for purposes where error checking and correction are either not necessary or are performed in the application. Time-sensitive applications (such as such as video playback or DNS lookups) often use UDP because dropping packets is preferable to waiting for packets delayed due to retransmission.

### 1.9.4 Application Layer Protocols

At the Application Layer, the following protocols are commonly used for IoT devices.

**1. HTTP**

The Hypertext Transfer Protocol (HTTP) is an application protocol used to transfer data on distributed and connected systems. This is what you use on a browser when you need to browse a website or a web application. HTTP is a stateless protocol, which means the client and web server make and break a connection for each interaction.

When a user requests to view a web page, that web server finds the requested web page, presents it to the user, and then terminates the connection. If the user requests a link within the newly received web page, a new connection must be set up, the request goes to the web server, and the web server sends the requested item and breaks the connection again. The web server never "remembers" the users who ask for different web pages, because it would have to commit a lot of resources for saving the state. HTTP generally works over port 80. Its specification is described in RFC 2616.

### 2. HTTPS

HTTPS is the secure version of HTTP. The 'S' at the end of HTTPS stands for 'Secure'. It means that all the communications between your client (browser, mobile apps) and the server (website, web application) is encrypted. HTTPS is often used to protect confidential online interactions such as online banking. Unlike HTTP, HTTPS is a stateful protocol where the session between the client and the server remains established until terminated (either by client or the server). You should prefer to use HTTPS over HTTP wherever possible for security and privacy protection. HTTPS generally works over port 443. Its specification is described in RFC 2818.

### 3. CoAP

The Constrained Application Protocol (CoAP) is a specialised web transfer protocol for use with constrained nodes (e.g., low-power, low-processing capabilities) and constrained (low-rate, error-prone, lossy) networks. The nodes often have 8-bitmicrocontrollers with small amounts of ROM and RAM, while constrained networks such as IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs) often have high packet error rates and a typical throughput of 10s of Kbit/s. The protocol is designed for machine-to-machine (M2M) applications such as smart energy and building automation. CoAP provides a request/response interaction model between application endpoints, supports built-in discovery of services and resources, and includes key concepts of the Web such as URIs and Internet media types. CoAP is designed to easily interface with HTTP for integration with the Web while meeting specialised requirements such as multicast support, very low overhead, and simplicity for constrained environments. Its specification is described in RFC 7252.

CoAP, although inspired by HTTP, was designed to use UDP instead of TCP. The message layer of CoAP over UDP includes support for reliable delivery, simple congestion control, and flow control. Some environments benefit from the availability of CoAP carried over reliable transports such as TCP or Transport Layer Security (TLS). CoAP over TCP, TLS, and WebSockets is described in RFC 8323.

### 4. WebSockets

Historically, creating web applications that need bidirectional communication between a client and a server (e.g., instant messaging and gaming applications) has required an abuse of HTTP to poll the server for updates while sending many distinct HTTP calls to the server. This results in a variety of problems such as

- The server is forced to use a number of different underlying TCP connections for each client: one for sending information to the client and a new one for each incoming message.
- Communication has high overhead as each client-to-server message requires an HTTP header.
- The client-side script is forced to maintain a mapping from the outgoing connections to the incoming connection to track replies.

A simpler solution could be to use a single TCP connection for traffic in both directions. This is what the WebSocket Protocol provides. Combined with the WebSocket API [WSAPI], it provides an alternative to HTTP polling for two-way communication from a web page to a remote server. The WebSocket Protocol enables two-way communication between a client to a remote server. It is based on TCP. The goal of this technology is to provide a mechanism for applications that need two-way communication with servers that does not rely on opening multiple HTTP connections.

The WebSocket protocol is commonly used for a variety of web applications such as games, stock tickers, multiuser applications with simultaneous editing and user interfaces exposing server-side services in real time. The WebSocket protocol is described in RFC 6455.

**5. MQTT**

Message Queuing Telemetry Transport (MQTT) is an open OASIS and ISO standard. It is a client-server publish and subscribe messaging transport protocol. It is light weight, open, simple, and designed to be easy to implement. These characteristics make it ideal for use in many situations, including constrained environments, such as for communication in Machine to Machine (M2M) and Internet of Things (IoT), where a small footprint is required and where network bandwidth is a challenge. The protocol runs over TCP/IP, or over other network protocols that provide ordered, lossless, bi-directional connections. It provides

- Publish and subscribe capability using which applications can subscribe to the server that publishes (generates and distributes) messages (data). Using publish and subscribe distribution model, the applications need not poll the server constantly for messages. Whenever the server has a message that needs to be distributed, it publishes the message and the subscribed applications receive it. Treat it like regular SMS that you get from your telephone provider or any other marketing company. If you gave them your number, they keep pushing messages for new deals or announcements without you asking for them.
- IoT devices could publish the messages (sensor data, readings, etc.) to the subscribed applications (or application servers).
- A messaging transport that does not depend upon the content of the payload (message or data).
- The three qualities (levels) of service for message delivery.
  - "At most once": At this level, messages are delivered according to the best efforts of the operating environment. Messages could be lost. You could use this level, for example, with sensor data where it does not matter much if an individual reading is lost as the next one will be published soon after.
  - "At least once": At this level, messages are sure to arrive, but duplicates can occur. You would require a mechanism in your application to handle the duplicate messages.
  - "Exactly once": At this level, messages are assured to arrive exactly once. This level could be used, for example, with billing systems where duplicate or lost messages could lead to incorrect charges being applied.
- A small transport overhead and protocol exchanges minimised to reduce network traffic.
- A mechanism to notify interested parties when an abnormal disconnection occurs.

**6. XMPP**

The Extensible Messaging and Presence Protocol (XMPP) enables the near-real-time exchange of structured yet extensible data between any two or more network entities. The purpose of XMPP is to enable the exchange of relatively small pieces of structured data (called "XML stanzas") over a network between any two (or more) entities. XMPP is typically implemented using a distributed client-server architecture, wherein a client needs to connect to a server in order to gain access to the network and thus be allowed to exchange XML stanzas with other entities (which can be associated with other servers). XMPP allows both client to server as well as server to server communication. XMPP could be used for communication amongst IoT devices. XMPP works over TCP. XMPP is described in RFC 6120.

**7. DDS**

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

In a distributed system, middleware is the software layer that lies between the operating system and applications. It enables the various components of a system to communicate and share data more easily. It simplifies the development of distributed systems by letting software developers focus on the specific purpose of their applications rather than the mechanics of passing information between applications and systems. DDS provides machine-to-machine communication. Like MQTT, DDS uses publish-subscribe model as well. Publishers (devices that generate and distribute data) create topics to which Subscribers (devices that consume the distributed data) can subscribe to. Applications communicate by publishing and subscribing to topics identified by their topic name. Subscriptions can specify time and content filters and get only a subset of the data being published on the topic. Using DDS, the data can also be shared with flexible Quality of Service (QoS) specifications including reliability, system health (liveliness), and even security.

**8. AMQP**

Advanced Message Queuing Protocol (AMQP) is an open OASIS and ISO standard for passing business messages between applications or organizations. It connects systems, feeds business processes with the information they need and reliably transmits onward the instructions that achieve their goals.

AMQP connects across

- Organisations – applications in different organisations
- Technologies – applications on different platforms
- Time – systems don't need to be available simultaneously
- Space – reliably operate at a distance, or over poor networks

AMQP is comprised of several layers. The lowest level defines an efficient, binary, peer-to-peer protocol for transporting messages between two processes over a network. Above this, the messaging layer defines an abstract message format, with concrete standard encoding. Every compliant AMQP process must be able to send and receive messages in this standard encoding.

## 1.10 Logical Design of IoT

The logical design of IoT refers to an abstract representation of the entities and processes that are involved in an IoT system. An IoT system comprises of a several functional blocks that provide the system the various capabilities for identification, sensing, actuation, communication, and management. Let's understand these logical components in detail.

### 1.10.1 IoT Functional Blocks

At a high-level, the logical representation of an IoT system could involve the following functional blocks. Each block could have its own set of processes, mechanisms, and protocols. These functional blocks work in an integrated way to provide an overall IoT system and its experience. This Fig. 1.10.1 may look very similar to one that you read in the Section 1.4" Basic Nodal Capabilities". It is just little more abstract over here.

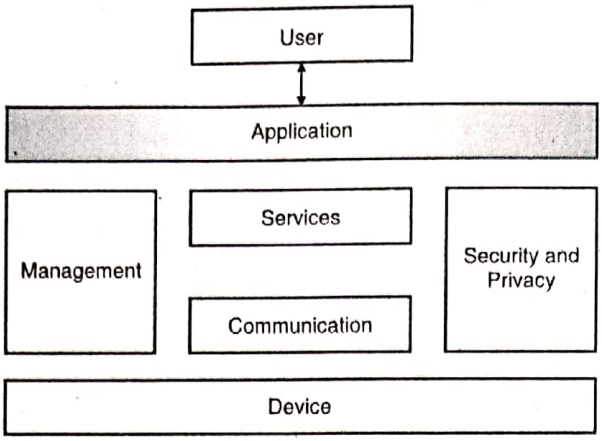



**Fig. 1.10.1 : IoT Functional Blocks**

1. **Application :** This is the topmost view of an IoT system. IoT application provides the interface using the user (or other systems) can potentially interact with the IoT device.

   Using the application, the user can provide commands (inputs), interact (consume) with the device output, and also configure it as desired. For example, using a smart bulb application, you could control which bulbs to power on or off, the luminosity (brightness) of each bulb and perhaps set a schedule when the bulbs should be automatically powered on or off.

2. **Device :** This is the bottom most view of an IoT system. It is the actual IoT device (or a set of IoT devices) that has been deployed to achieve a particular purpose. For example, the deployed IoT device could be a thermostat or a virtual assistant that could take commands from the user. The device could have sensors, actuators, or both.

3. **Communication :** Communication is what connects and adds the device to the network. There could be several communication protocols and mechanisms that a device could support. You now know quite a few of them already from our previous discussions.

4. **Services :** This block represents other (non-communication oriented) services, mechanisms and protocols that could be used for functions such as device monitoring, health reporting, device controlling, device data publishing and device discovery.

5. **Management :** Management block handles functions such as firmware update, patching, configuring the device for use, alerting the user when the device needs attention, and other maintenance and management activities.

6. **Security and Privacy :** Security and Privacy block provides various services such as identification, authentication, authorisation, data encryption, data integrity and device availability. It ensures that any unauthorised users or systems cannot interact with the IoT device and there is appropriate protection for the data either when it is received or sent from the device.

## 1.11 IoT Communication Models

At a high-level, Fig. 1.11.1 shows the major communication models used in IoT.

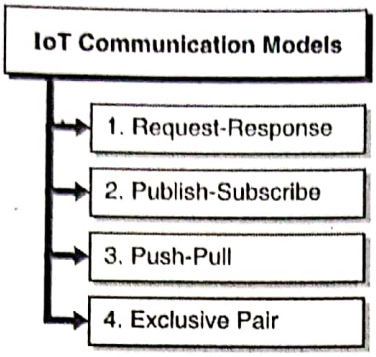



**Fig. 1.11.1 : IoT communication models**

### 1. Request – Response Communication Model

Request – Response is the simplest and oldest method of communication between devices in a network. At a high-level, it can be diagrammatically represented as shown in the Fig. 1.11.2.

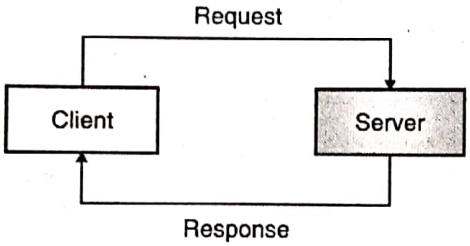



**Fig. 1.11.2 : Request-response Communication model**

The client typically sends the request to the server. The request contains the exact resource, data, or processing that the client expects the server to do and communicate back the response. The server "understands" the client request, does the required processing, gets the desired resources, forms a response, and sends it back to the requesting client. A simple example of request-response communication model could be you browsing an e-commerce portal or watching videos without login into your account.

Your client (browser) just establishes a connection with the server (website) and helps you to navigate different pages or watch videos. Note here that there could be several steps to be taken before the client and server can actually communicate and carry out a request-response communication.

For simplicity, those steps are not listed over here. Also, request-response is a stateless communication. The server does not maintain the track of previous requests from the client (Do not confuse this with transaction history or anything else. I am just talking about the communication model tracking previous request-response or not. The application can very well maintain all historical communication).

## 2. Publish – Subscribe Communication Model

You have, so far, already read about publish-subscribe model in MQTT and DDS. Let's understand it in further detail. Publish-subscribe messaging (or communication model), or pub/sub messaging, is a form of asynchronous service-to-service communication used in modern computing environments. In a publish-subscribe communication model, any message published to a topic is immediately received by all of the subscribers of the topic. Publish-subscribe communication model can be used to enable event-driven architectures, or to decouple applications in order to increase performance, reliability, and scalability. At a high-level, it can be diagrammatically represented as shown in Fig. 1.11.3.

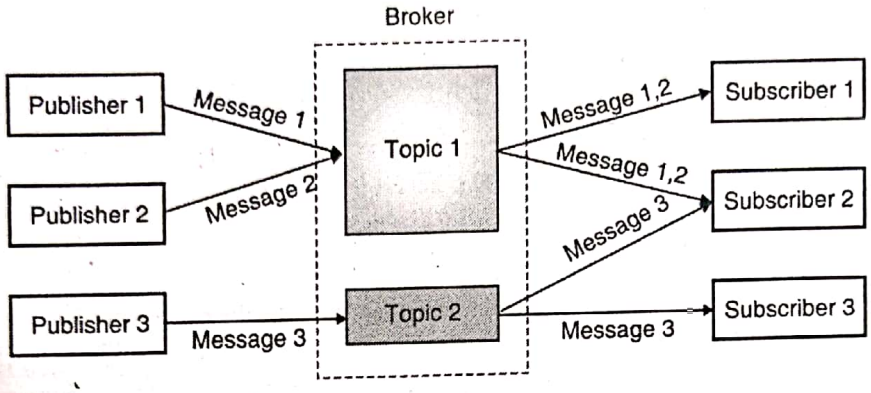



**Fig. 1.11.3 : Publish – Subscribe Communication Model**

**(a) Publisher :** Publisher is an entity that generates the messages (or data) and pushes them to a topic. A publisher need not be aware of all the subscribers for its messages that are pushed to topics.

**(b) Topic :** Topic is a logical entity that holds the messages sent by the publishers. A topic could receive messages from multiple publishers.

**(c) Subscribers :** Subscribers are the consumers that are interested to get messages (or data) on a particular topic without directly communicating with the publisher. A subscriber can subscribe to various topics.

**(d) Brokers :** There could be an optional third-party between publisher and subscriber called Broker. A Broker may hold the topics to which publishers can publish and subscribers can subscribe to.

By the way, have you ever subscribed to a YouTube channel that you wish to follow?

If yes, you understand what a subscription is.

- Channel owner is the publisher that releases various videos.
- Channel name is the topic under which the publisher publishes the new videos.
- YouTube is a broker that holds these published videos (and many others) and is aware of subscribers. You are a subscriber that has let known YouTube (the broker) to notify you when the publisher has published a new video on the channel (topic). You could have subscribed to one or many YouTube channels.

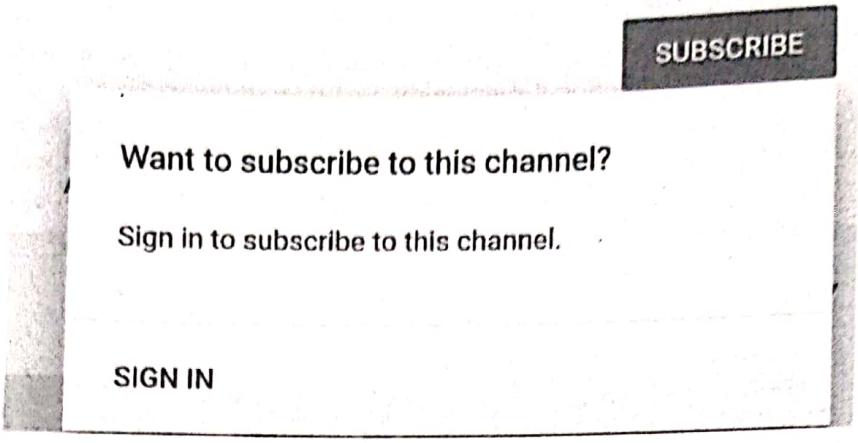



**Fig. 1.11.4**

### 3. Push – Pull Communication Model

At a high-level, the push-pull communication model can be diagrammatically represented as shown in Fig. 1.11.5.

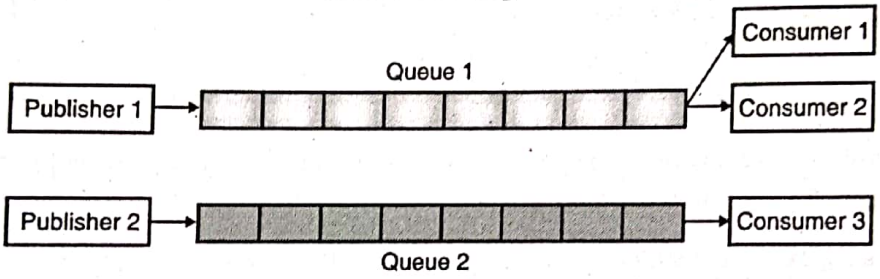



**Fig. 1.11.5**

In the push-pull communication model, the data publishers push the data elements (messages) into logical entities called message queues. The consumers then pull the data elements (messages) from the message queue and process (consume) it as desired. The messages are usually small, and can be things like requests, replies, error messages, or just plain information. Many publishers and consumers can use the queue, but each message is processed only once, by a single consumer. For this reason, this messaging pattern is often called one-to-one, or point-to-point communications. AMQP and RabbitMQ are the commonly used message queuing frameworks. Message queue based communication model is a form of asynchronous service-to-service communication used in modern computing environments. It can be used to enable event-driven architectures, or to decouple applications in order to increase performance, reliability, and scalability.

Messages are stored on the queue until they are processed and then deleted. Each message is processed only once, by a single consumer (there could be several consumers that might be pulling messages from the same queue. The point is that each message is pulled and consumed by only one consumer irrespective of how many consumers a queue has). For example, assume that there is a single big queue of people buying movie tickets but there are multiple counters selling the tickets. People from front of the queue keep going to different counters as their turns come. Each person visits a counter only once and is served. Queues also act as a buffer which helps in situations where publishers are pushing data at a faster rate than the rate at which the consumers can consume it. True, isn't it ? A queue, at the end of the day, is a buffer to hold people until their turn comes in.

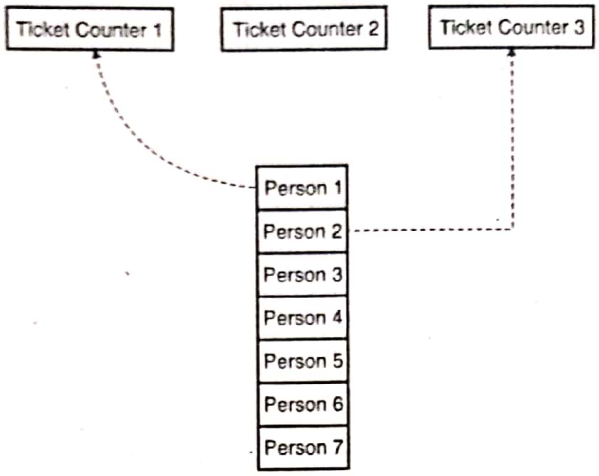



**Fig. 1.11.6**

Hundreds of people may join the queue for buying tickets or a new phone model, but they are served one by one. Message queues might be of different types. Some of the common ones are as following.

**1. Based on delivery**

(a) At-Least-Once : A message is delivered at least once, but occasionally more than one copy of a message is delivered.

(b) Exactly-Once : A message is delivered once and remains available until a consumer processes and deletes it. Duplicate messages are not introduced into the queue.

**2. Based on ordering**

(a) Best-Effort Ordering : Occasionally, messages might be delivered in an order different from which they were sent.

(b) First-In-First-Out : The order in which messages are sent and received is strictly preserved (i.e. First-In-First-Out).

**4. Exclusive Pair Communication Model**

Exclusive Pair communication model is an extension of client-server communication model where the server keeps the connection open for further requests from the same client instead of closing it after every response. It is a stateful communication where prior requests and responses are "remembered" until the connection is closed.

At a high-level, the exclusive pair communication model can be diagrammatically represented as shown in the Fig. 1.11.7.

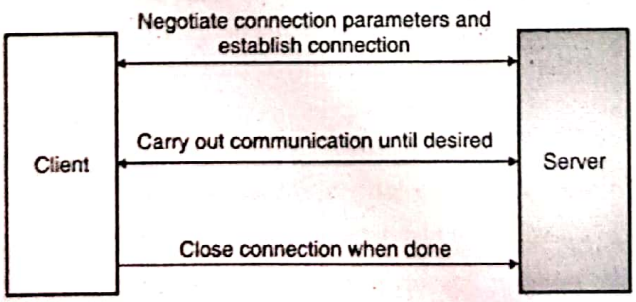



**Fig. 1.11.7 : Exclusive pair communication model**

This type of communication model is typically used for instant messaging or online games. Both, the client, and the server can send each other messages without re-establishing the connection after the initial connection has been established. The session is preserved until it is no more desired.

### 1.11.1 Comparison between IoT Communication Models

The Table 1.11.1 summarises the comparison between various IoT communication models.

**Table 1.11.1**

| Comparison Attribute | Request-Response | Publish-Subscribe | Push-Pull | Exclusive Pair |
|---|---|---|---|---|
| State | Stateless | Stateless | Stateless | Stateful |
| Client request / action | Required | Not required | Required | Optional |
| Type of communication | Synchronous | Asynchronous | Asynchronous | Synchronous |
| Resource requirement | Low | Low | Low | High |
| Connection Overhead | High | Low | Low | Low |
| Scalability | Medium | High | High | Low |
| Direct connectivity between server and client | Required | Not required | Not required | Required |
| Message access and distribution | One-to-one | One-to-many | One-to-many | One-to-one |

## 1.12 IoT Communication APIs

Now, that you have a good understanding of the various communication models, let's learn about the two most commonly used APIs for IoT communication – REST-based and WebSocket-based.

### 1.12.1 REST-based Communication APIs

#### 1. REST

REST is an acronym for Representational State Transfer.

✍ ***Definition : REST is an architectural style for distributed hypermedia systems describing the software engineering principles and interaction constraints.***

REST is extensively used today to design web services. REST defines how system resources can be addressed and transferred (between server and client) using various methods. REST is based on request-response communication model. All REST interactions are stateless. Let's understand and define a few key terms as you understand more about REST.

#### 2. Resource

✍ ***Definition : The target of a client request is called a Resource.***

REST does not limit the nature of a resource. It just defines an interface that might be used to interact with resources.

Each resource is identified by a Uniform Resource Identifier (URI). Any information that can be named can be a resource. A resource could be a web page element, a video, an audio, a document, or anything else. REST uses a resource identifier to identify the particular resource involved in an interaction between components. When a client constructs a request message, it sends the target URI in the request. When a request is received, the server reconstructs an effective request URI for the target resource.

### 3. Representations

*Definition : A representation is information that is intended to reflect a past, current, or desired state of a given resource.*

It is in a format that can be readily communicated via a protocol. For example, it could be a HTML document or a JPEG image. REST components perform actions on a resource by using a representation to capture the current or intended state of that resource and transferring that representation between components. A representation is a sequence of bytes, plus representation metadata to describe those bytes (how those bytes should be interpreted).

### 4. Connectors

REST uses various connector types to encapsulate the activities of accessing resources and transferring resource representations. The connectors present an abstract interface for component communication. A connector manages network communication for a component and hence any information can be shared across multiple interactions in order to improve efficiency and responsiveness. Some of the common connectors could be as shown in Table 1.12.1.

**Table 1.12.1**

| Connector | Modern Web Examples |
|---|---|
| client | libwww, libwww-perl |
| server | libwww, Apache API, NSAPI |
| cache | browser cache, Akamai cache network |
| resolver | bind (DNS lookup library) |
| tunnel | SOCKS, SSL after HTTP CONNECT |

### 5. Request Methods

Request method indicates the purpose for which the client has made the request and what is expected by the client as a successful result.

Some of the commonly used standardised methods are as shown in Table 1.12.2.

**Table 1.12.2**

| Method Name | Description |
|---|---|
| GET | Transfer a current representation of the target resource. GET is the primary mechanism of information retrieval. Hence, when you speak of retrieving some identifiable information via HTTP, you are generally referring to making a GET request. |
| HEAD | Same as GET, but only transfer the status line and header section without the message body. The server sends the same header fields in response to a HEAD request as it would have sent if the request had been a GET, except that the payload header fields may be omitted. |
| POST | Perform resource-specific processing on the request such as sending a message, creating a new resource, etc. |

| Method Name | Description |
|---|---|
| PUT | Replace all current representations of the target resource with the representation in the request. |
| DELETE | Remove all current representations of the target resource |
| CONNECT | Establish a tunnel to the server identified by the target resource |
| OPTIONS | Describe the communication options for the target resource |
| TRACE | Perform a message loop-back test along the path to the target resource |

## 1.12.1(A) REST Architectural Constraints

REST defines several architectural constraints (principles). They are as shown in Fig. 1.12.1.

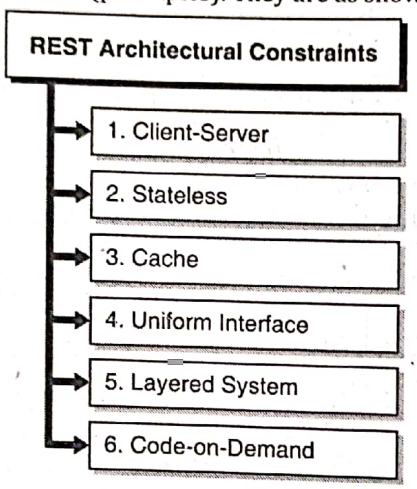



**Fig. 1.12.1 : REST Architectural Constraints**

1. **Client-Server** : The Client-Server architectural constraint ensures the separation of responsibilities (separation of concerns). The client does not need to worry about the server and its resources and similarly the server need not worry about the client and its interfaces. This separation of concerns allows both client and server to be independently developed, managed, updated, and scaled without impacting each other's functionality.
2. **Stateless :** The Stateless architecture constraint ensures that the communication is stateless in nature which means that each request from client to server must contain all of the information necessary to understand the request and cannot take advantage of any stored context (or previous requests) on the server.

   The session state is, therefore, kept entirely on the client. This constraint provides the following benefits.

   (a) Improved visibility : Visibility is improved because a monitoring system does not have to look beyond a single request in order to determine the full nature of the request. For example, a firewall could block the request as it entirely understands what the request is about.

   (b) Improved reliability : Reliability is improved because it eases the task of recovering from partial failures.

   (c) Improved scalability : Scalability is improved because not having to store state between requests allows the server component to quickly free resources, and further simplifies implementation because the server doesn't have to manage resource usage across requests.

3. **Cache :** The Cache architecture constraint ensures that a client could reuse the server response data later (for the same of equivalent requests) to improve network efficiency and performance. This, however, requires that the data within a server's response to a request is implicitly or explicitly labelled as cacheable or non-cacheable.
4. **Uniform Interface :** The Uniform Interface architecture constraint ensures that the method of communication between the REST components (say client and server) must be uniform such that the implementations are decoupled from the services they provide. The information is transferred in a standardised form rather than the one which is specific to an application's needs. REST is defined by four interface constraints - identification of resources, manipulation of resources through representations, self-descriptive messages, and hypermedia as the engine of application state.
5. **Layered System :** The Layered System architecture constraint allows an architecture to be composed of hierarchical layers by constraining component behaviour such that each component cannot "see" beyond the immediate layer with which it is interacting. By restricting knowledge of the system to a single layer, you can avoid the overall system complexity and promote architectural independence amongst the REST components.

   For example, to improve performance and distribute server load, you can place a load balancer in front of several servers. A client request then can go to the load balancer instead of a server directly. A client need not know this or do anything specific to interact with the load balancer. The placement of an intermediate layer, such as a load balancer, is completely transparent to the client. Hence, you can place any intermediate layers between the REST components and improve system scalability across multiple networks and processors.
6. **Code-on-Demand :** The Code-on-Demand architecture constraint allows the client functionality to be extended by downloading and executing code in the form of applets or scripts from the server. This simplifies clients by reducing the number of features required to be pre-implemented. Allowing features to be downloaded after deployment improves system extensibility. This is an optional constraint.

### 1.12.1(B) RESTful Web Service (or APIs)

Alright, so now that you understand REST a bit better, let's come back and understand further that how REST-based Communication APIs work.

RESTful (or REST-based) APIs work with HTTP according to REST architectural constraints (principles) using the various request methods such as GET and POST.

There could be several REST components, such as client and server, in the communication path. Each resource has a specific identifier defined by its URI and has a representation associated with it.

RESTful APIs can use various media types for submitting requests or providing responses. JSON and XML are most commonly used media types for working with RESTful APIs or web services.

The Fig. 1.12.2 provides a high-level view of REST-based Communication APIs.

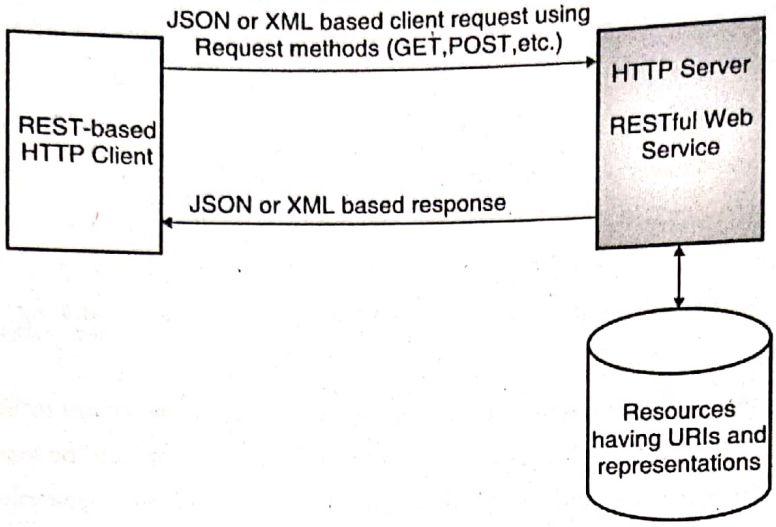



**Fig. 1.12.2 : REST-based Communication APIs**

## 1.12.2 WebSocket-based Communication APIs

WebSocket APIs allow bidirectional, full duplex communication between clients and servers. WebSocket APIs follow the exclusive pair communication model. You have already learnt about the WebSockets protocol in the Section 1.7.4 "Application Layer Protocols".

The Fig. 1.12.3 provides a high-level overview of how WebSockets APIs establish the connection and then carry out bidirectional data transfer between the client and the server. The connection is closed when further data transfer is not desired. Note here that the connection is established only once, and it is kept alive until the data transfer is complete unlike request-response protocol where the connection is established on every request and is terminated after receiving the response.

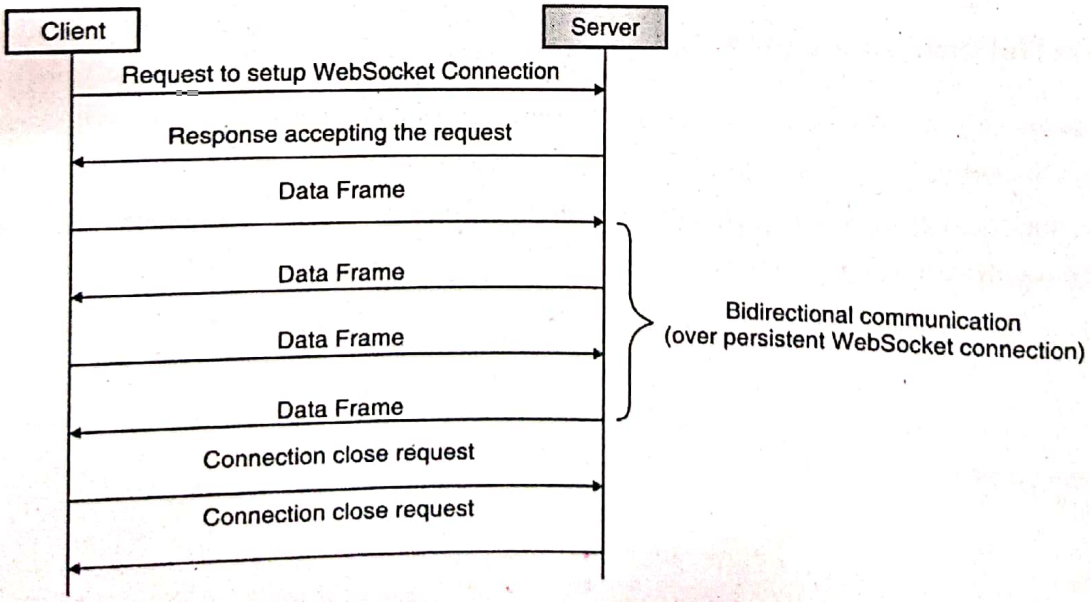



**Fig. 1.12.3 : WebSocket APIs**

In the WebSocket Protocol, data is transmitted using a sequence of data frames. To avoid confusing network intermediaries (such as intercepting proxies) and for security reasons, a client masks all frames that it sends to the server. The server closes the connection upon receiving a frame that is not masked. However, a server does not mask any frames that it sends to the client. A client closes the connection if it detects a masked frame from the server.

### 1.12.3 Comparison between REST-based and WebSocket-based APIs

The Table 1.12.3 summarises the comparison between REST-based and WebSocket-based APIs.

**Table 1.12.3**

| Comparison Attribute | REST-based | WebSocket-based |
|---|---|---|
| Communication Model | Request-Response | Exclusive Pair |
| Data Flow | Server responds to the client on each request (unidirectional) | Bidirectional |
| State information | Not preserved (stateless) | Preserved (stateful) |
| Session maintained by | Client | Server and Client |
| Resource requirements | Comparatively lower | Comparatively higher |
| Connection overhead | Required | Not required |
| Uses | Get or process data | Real-time applications such as games |
| Scalability | High | Low |

## 1.13 IoT Enabling Technologies

IoT systems depend upon several other technologies and mechanisms for enabling them to function as desired. Some of the key technologies are as shown in Fig. 1.13.1. I will touch upon in brief about them.

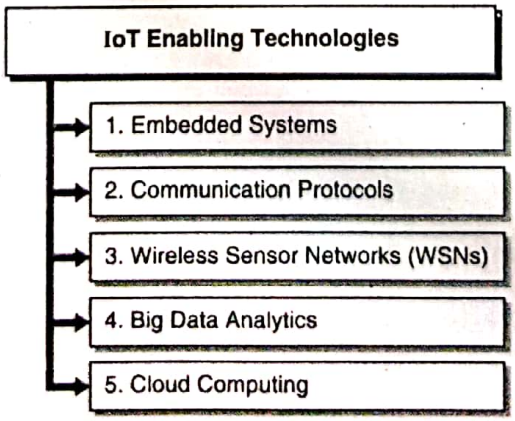



**Fig. 1.13.1 : IoT Enabling Technologies**

## 1.14 Embedded Systems

The dictionary meaning of the word embedded is "enclosed closely in". You would have come across the word embedded systems quite a few times. What picture comes into your mind when you hear about it? Do you visualise a circuit board? Do you visualise an old computer? Do you visualise a mini system of some sort hanging by a device?

Let's understand what embedded systems are.

✍ ***Definition :*** *Embedded systems are specialised electronic systems that carry out limited and specific functionalities.*

- They have special purpose hardware and special purpose software burnt into ROM as firmware that carries out very specifically programmed tasks. Usually they are part of a larger machinery or larger ecosystem of various units that work in tandem to do something meaningful.
- A schematic diagram to think of an embedded system could be as shown in Fig. 1.14.1.

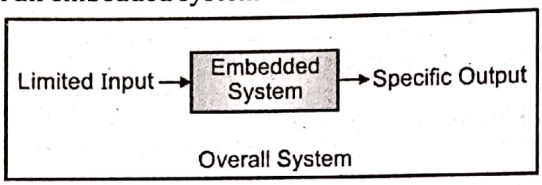



**Fig. 1.14.1 : Schematic diagram of Embedded System**

- Note here that general purpose computers such as Smartphones, tablets, laptops, desktops, and other similar forms of general purpose computing are excluded from the list of embedded systems because they do not have a fixed task and can perform various tasks as programmed.
- Embedded systems carry out very specific tasks such as turning off the refrigerator compressor when the desired temperature is reached or like blinking a red LED if the car door is open while driving. Do not, however, assume that the embedded system cannot have a keyboard or mouse or a display unit such as a monitor. Just keep in mind that the embedded systems are very specific task-oriented.

**Note :** Even embedded system experts often argue about what systems can be considered embedded system and what not. A famous debate is around tablets. Tablets are becoming more general purpose where you can install apps and carry out nearly everything that a laptop or a desktop could do. Hence, by the definition and working principles of an embedded system, they cannot be considered to be embedded systems. However, embedded systems have other embedded system characteristics such as limited memory, processing power etc. which make people argue that it should be considered embedded system. But, for that matter, even a laptop has limited memory and capabilities and hence that argument does not hold true. I would strongly NOT consider modern tablets to be an example of embedded systems and would rather call them general purpose computing devices. An embedded system MUST BE specific task-oriented only in the first place before even thinking about and evaluating any other embedded system characteristics.

### 1.14.1 Comparison between Embedded Systems and Other Computing Systems

- The Table 1.14.1 summarises the key comparison parameters between embedded systems and other general purpose computing systems.

**Table 1.14.1 : Comparison between Embedded Systems and other computing systems**

| Comparison Attribute | Embedded Systems | Other Systems |
|---|---|---|
| Tasks carried out | Very specific | General purpose |
| Form factor | Usually small | Usually not small |
| OS | Typically burnt as firmware, not easily replaceable | Installed in hard drive, easily replaceable |
| Directly operated by | Larger systems, machines, or units | End users (human) |
| Maintenance activities | Less frequent | Frequent |
| Power consumption | Low | High |
| Reliability | High | Low |
| Externally visible | Usually no | Usually yes |
| Programmed by | Microcontroller engineers | Software developers |
| Interaction mechanism | Usually buttons, LED, sounds, heat, etc. | Display, touch, mouse, keyboard, etc. |

## 1.14.2 Applications (Domain) of Embedded Systems

- Embedded systems are around us so much so as to make it really difficult to experience and imagine life without them.

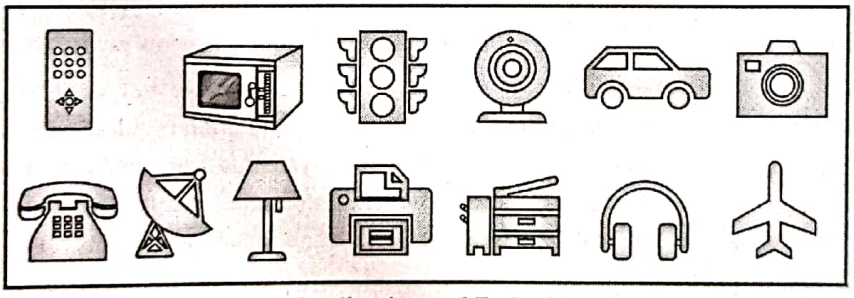


**Fig. 1.14.2 : Applications of Embedded Systems**

- Some of the common applications of embedded systems are as following.
  1. **Household equipment :** Most of the household electronic equipment have some sort of embedded system circuitry that makes them operational. These could be microwave, air-conditioners, refrigerators, washing machines, tv, music player, disc player, LED lighting, set-top boxes, casting devices, remote control, decoration lamps, geysers, etc.
  2. **Industrial equipment :** Various industries use respective equipment that have embedded systems. It could range from aircraft controllers, scanners, X-ray machines to automation systems, humidity controllers, fire alarms, smoke detectors, and robots.
  3. **Other equipment :** There could be other equipment such as calculators, mouse, currency note counters, cameras, elevator controlling system, printer, copier machines, wireless phones, car controlling system, drones, toys, etc. that heavily use embedded systems to carry out their functionalities.

## 1.14.3 Characteristics of Embedded Systems

The major characteristics of embedded systems are shown in Fig. 1.14.3.

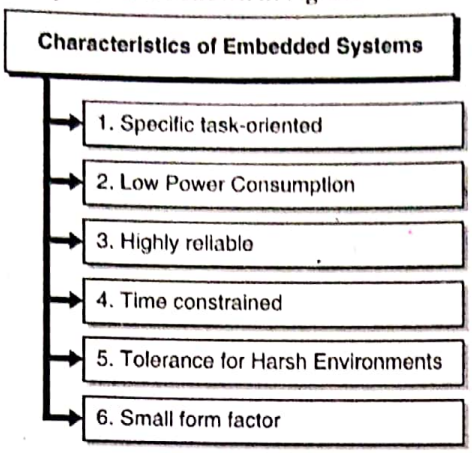



**Fig. 1.14.3 : Characteristics of Embedded Systems**

1. **Specific task-oriented :** By now, I have told you this enough number of times already. One of the key characteristics of an embedded system is that it exists to solve a particular and specific task only that is carefully programmed and burnt into its firmware. You just cannot plug out an embedded system from a refrigerator and plug it into washing machine and make it to work. Embedded systems have limited, and specific functionalities developed specifically for the desired target systems.
2. **Low Power Consumption :** Quite a few embedded systems such as camera, toys, remote controls, work on battery power. Hence, it is important that they consume low power to avoid quickly draining out batteries. Even though for some embedded systems it might look like that they run on external power, it may not be so. The larger system could work on external power, but the embedded system could have its own separate battery source that keeps it operative even during power disruption. Hence, it is desired that embedded systems have low power consumption.
3. **Highly reliable :** Embedded systems must be highly reliable as they are not easily replaceable or regularly maintainable. It is difficult to take them out from the larger systems and often requires factory equipment and highly skilled technicians to fix them when broken. Replacing embedded system is costly and requires high downtime for the equipment (or the larger systems that they make functional). Sometimes, it might be the case that the embedded system is no more available or is obsolete in which case the entire equipment needs to be replaced.
4. **Time constrained :** Embedded systems must respond within strict time limits. Imagine you pressing a button on remote and it takes 2 minutes to switch the channel or increase the volume as the desired effect. Not acceptable right? Similarly, assume that there is a fire alarm or smoke detector. You would want them to alert as soon as possible rather than when the damage is already done. Hence, embedded systems are time constrained and must be responsive within the appropriate time limit.
5. **Tolerance for Harsh Environments :** Embedded systems are often used in extreme temperatures (think near car engines or aircrafts, as fire alarms, etc.), sounds, shocks, and other similar rough environments in the industrial as well as in the general usage circumstances. They should be able to tolerate such harsh environments and should continue to operate within the prescribed tolerance limits as required for target systems.
6. **Small form factor :** Embedded systems are often part of larger systems. They are usually small such that they can be fitted into systems without occupying a lot of space. They have limited processing power, limited memory, minimal peripherals lots, and limited extensibility. They generally allow plugging-in external devices such as a keyboard or a display as and when required for diagnostic or programming purposes. Additionally, they are built with minimal and just enough features to keep the cost low and are usually mass produced.

### 1.14.4 Communication Protocols

As you understand, communication protocols are the ones that actually enable IoT devices to connect to a network. You have already learnt about several communication protocols in the Section 1.7 "IoT protocols". Refer to that section for a refresher. I will not duplicate the information again here.

### 1.14.5 Wireless Sensor Networks (WSNs)

Several sensor devices can be connected together over a network.

✍ ***Definition :*** *Wireless Sensor Network (WSN) is a large collection of sensor devices that can monitor several physical conditions.*

Each sensor device is called a sensor node. A sensor node can monitor several physical conditions such as temperature, air pressure, illumination of light, movement of people, wind speed, humidity, etc.

The collected information is sent to the processing center via the WSN gateway. The processing center evaluates the information received from the various sensor nodes and then sends the instructions to the connected devices to act suitably. For example, if the processing center finds, from the data received from WSN, that the intensity of light is too high, it can instruct the bulb (connected device) to reduce its illumination to adjust to a particular intensity level.

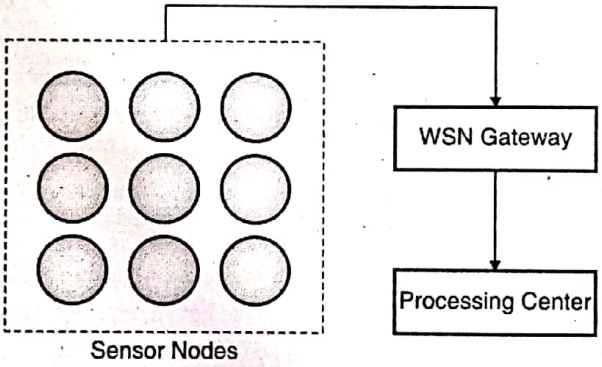



**Fig. 1.14.4 : Wireless Sensor Networks (WSNs)**

### 1.14.6 Big Data Analytics

Increasingly people and things are getting interconnected. Data is continuously created by devices and users. For example, when you go for online shopping all your clicks and views, your interaction with the website, your interaction with the competitor website, your addition to the cart, removal from the cart, price comparison, review reading, etc. are all recorded to analyse you as a user and a prospective buyer who could be influenced to make a purchase. The entire agenda of conducting data analytics is based on making informed decisions that can be further used to shape your behaviour and drive the business intentions. Similarly, the amount of data that IoT devices collect could be analysed to find patterns and take actions.

The term Big Data (always write capitalised B and D in Big Data. Big Data is a noun that has a special meaning!) refers to an accumulation of data that is too large and complex for processing by traditional database management tools. The datasets referred under Big Data are massive and could be petabytes in size.

✍ ***Definition :*** *Big Data consists of extensive datasets that require a scalable architecture for efficient storage, manipulation, and analysis.*

Big Data problems usually require specific set of tools and techniques for processing and usually are not traditional database systems. The data to be processed could be historical or could be accumulated in real time.

## 1.14.6(A) The Five Vs (Characteristics) of Big Data

Big Data is often characterised as 5 Vs. They are as shown in Fig. 1.14.5.

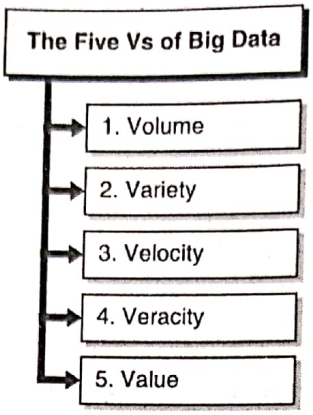



**Fig. 1.14.5 : The Five Vs (Characteristics) of Big Data**

1. **Volume :** Volume refers to the size of the dataset. It could consists of billions of rows and millions of columns. It typically ranges from terabytes to petabytes of data. For example, imagine a dataset of all active users on flipkart.com. It could be a listing of all users and the clicks they made in a day. Assuming a user showed interest in 5 different mobile handsets and clicked on 10 photos each it would mean 50 data points per user. Now, if flipkart.com has 1 million users looking at mobile handsets on a sale day, it could mean 50 million data points!
   Usually such datasets are stored in multi-tiered storage media. The data storage and retrieval could be time consuming.
2. **Variety :** Variety refers to the data accumulated from multiple data sources. There could be various types of data collected in various formats and structures. The data could be structured, semi-structured or unstructured. For example, data on cricket world cup could mean scoring tables, videos, audio, tweets, texts, images, comments, and news reports. Such a wide variety of data on a particular topic could make Big Data analysis further complex. The data is carefully chosen so as to be meaningful for the purpose of analysis and objectives to be met.
3. **Velocity :** Velocity refers to the rate of speed at which the new data is generated and processed. It could be generated in real time or could be historical in nature. For example, in a live match, several new tweets could be generated in a second. If you were to write a Big Data application that analyses tweets in real time, it could mean ingesting tweet data at the rate of millions per second. Many Big Data applications process real time data such as "likes" and "shares" to report trends and top of the hour news. But, keep in mind that Velocity may not always be high. For example, medical records could have been compiled over several years before they are analysed.
4. **Veracity :** Veracity refers to the integrity of the Big Data. It is a measure of data quality and usefulness of the data. For example, a dataset on cancer patients could have low veracity if it has data for non-cancer patients as well. Such a data could produce inaccurate insights and may not be useful for carrying out Big Data analysis. You should ensure that the data chosen for Big Data analysis is accurate and is free from biases, noises, and abnormality.
5. **Value :** Value measures the degree of usefulness of data for the organisation. The longer it takes for the data to generate meaningful results, the lower the value of the data. In a way, Value of data is dependent on the Veracity of data. High Veracity data generally has high value for the organisation. Value of data also depends upon
   (a) How it was stored
   (b) How old / recent it is
   (c) How the data attributes were preserved
   (d) What questions can be answered with the data

## 1.14.7 Cloud Computing (Interdependencies of IoT and Cloud Computing Systems)

✍ ***Definition :*** *Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.*

In a nutshell, the above definition means that the cloud computing is a service consumption mindset where you consume the various information technology services without having to own the hardware required for delivering them. For example, suppose you need 5 computers. Using the cloud computing model, you can get access to those 5 computers via a service provider over the internet. You would be charged for the time duration for which you use the computer.

Using the cloud computing technology scenario described above,

1. You did not have to buy the 5 computers yourself.
2. You were only charged for the duration you actually used those 5 computers.
3. Those 5 computers are accessible from anywhere (since they are provided over internet).
4. You don't really have to worry about power, network, disk, or other hardware and operational maintenance yourself.
5. Tomorrow, if you need another 5 computers, you can easily update your subscription and instantly get them.

Various cloud service providers provide several services for IoT. For example, the Fig. 1.14.6 shows is a snapshot from Amazon Web Services (AWS). It provides various IoT services on its cloud computing environment.

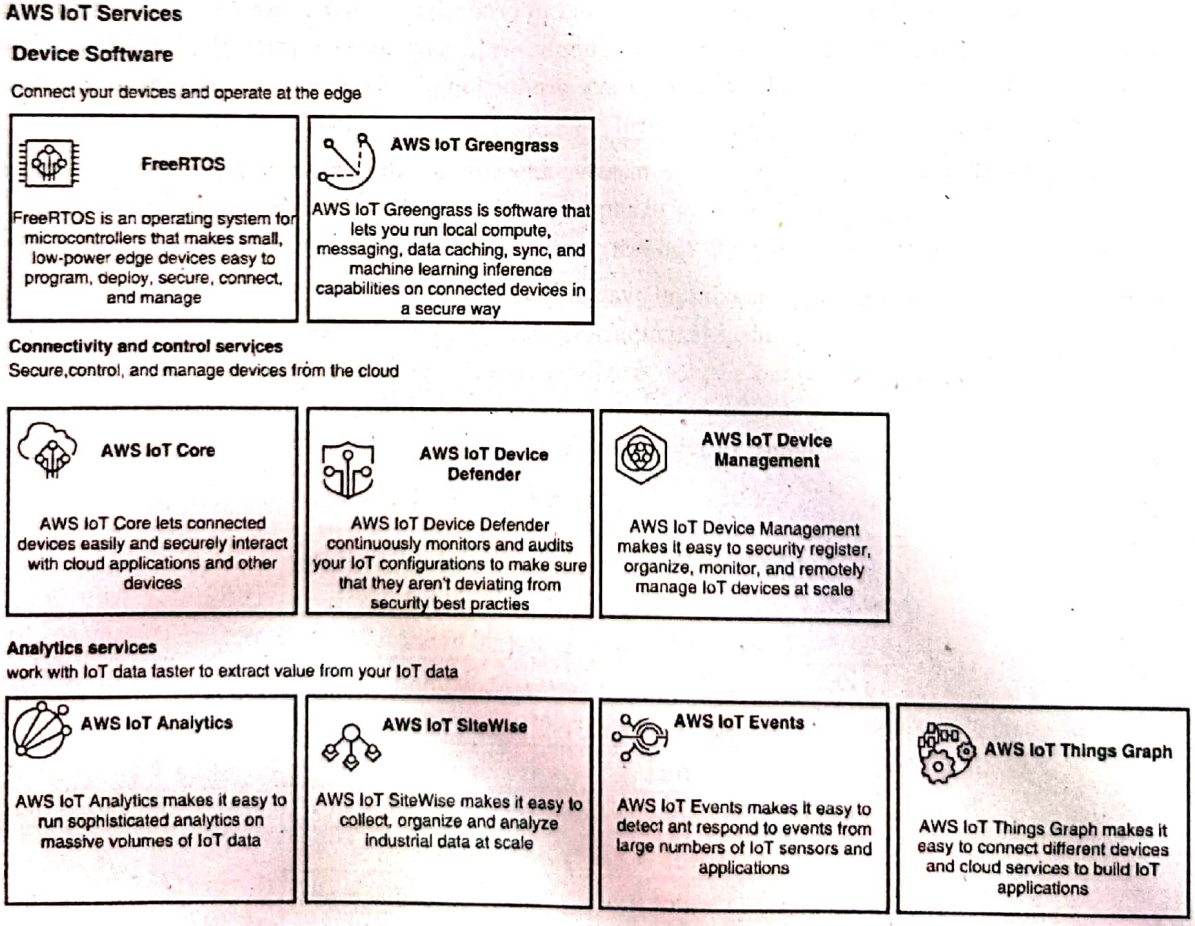



**Fig. 1.14.6 : AWS IoT Services**

## 1.14.7(A) Goals of Cloud Computing

Fig. 1.14.7 shows the high-level goals of cloud computing.

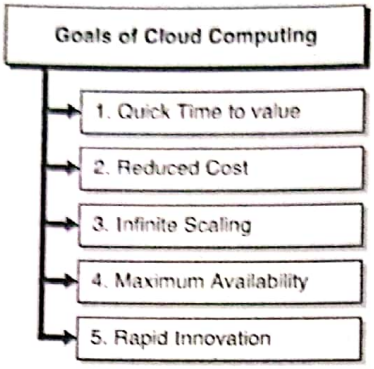



**Fig. 1.14.7 : Goals of Cloud Computing**

1. **Quick time to value :** Cloud computing allows you to quickly consume the IT resources and focus on key areas of your business without having to worry about managing the underlying infrastructure. For example, you do not have to worry about how electricity is produced. You just consume it for your purpose without having to think about electricity production.
2. **Reduced costs :** Without having to put capital investment in procuring the hardware and setting up the datacentre yourself (that requires managing facilities, staff, power supply, etc.), cloud computing enables you to pay for only what you use. For example, you do not have to setup electricity production plant yourself. You just consume electricity and pay for what you use based on the predetermined tariff.
3. **Infinite Scaling :** Cloud providers typically have massive amount of computing resources. These resources can be dynamically consumed based on your needs. For example, you don't really think from consumption perspective that how much electricity you are left with in-stock. You just go on using it assuming that it is infinitely available to you.
4. **Maximum availability :** Cloud provides maximum availability of computing resources. The resources are aggregated in a large resource pool. If any of the resource is temporarily out of service, it is easily replaceable by another resource without impacting the service. The cloud service providers typically provide Service Level Agreements (SLA) with respect to the services that they provide. The availability is generally referred in the industry in terms of "nines". Table 1.14.2 outlines what different "nines" of availability mean.

**Table 1.14.2**

| Availability | Annual Downtime |
|---|---|
| 99.9% ("three nines") | 8.77 hours |
| 99.95% ("three nines five") | 4.38 hours |
| 99.99% ("four nines") | 52.60 minutes |
| 99.995% ("four nines five") | 26.30 minutes |
| 99.999% ("five nines") | 5.26 minutes |
| 99.9999% ("six nines") | 31.56 seconds |
| 99.99999% ("seven nines") | 3.16 seconds |
| 99.999999% ("eight nines") | 315.58 milliseconds |
| 99.9999999% ("nine nines") | 31.56 milliseconds |

It is not unusual for a cloud service provider to provide an SLA of "five nines" which just means a downtime of 5.26 minutes in a year!

5. **Rapid innovation :** Cloud providers are rapidly innovating to bring new services and to deliver optimum performance at a cheapest possible cost. You directly benefit from any key technology breakthrough or any cost savings arising out of such innovation. For example, if a country moves from hydro-electric power plants to nuclear power plants, the benefits are automatically provided to its consumers. The consumers need not know how a nuclear reactor works for power generation. Similarly, new breakthroughs in technologies or better ways of computing are adopted by cloud service providers and you are benefitted automatically.

## 1.15 IoT Levels and Deployment Templates

So far you have learnt that typically an IoT system has the following components.

1. **IoT Device :** It allows identification, remote sensing, actuating and remote monitoring capabilities.
2. **Resources :** Resources are software components on the IoT device for accessing, processing, and storing sensor information, or controlling actuators connected to the device. Resources also include the software components that enable network access for the device.
3. **Controller Service :** Controller service runs on the device and interacts with the web services. Controller service sends data from the device to the web service and receives commands from the application (via web services) for controlling the device.
4. **Database :** Database stores the data generated by the IoT device.
5. **Web Service :** Web services serve as a link between the IoT device, application, database, and analysis components. Web service can either be REST-based on WebSocket-based.
6. **Analysis Component :** The Analysis Component is responsible for analysing the IoT data and generate results in a form which are easy for the user to understand and take insightful actions.

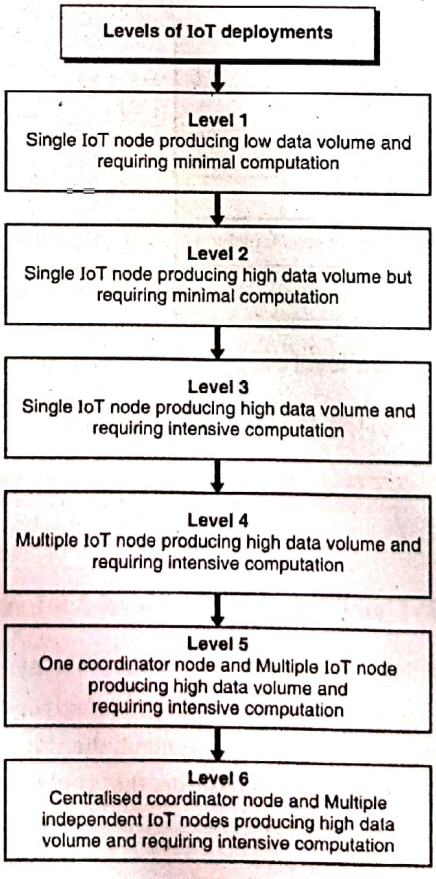



**Fig. 1.15.1 : Levels of IoT deployment**

7. **Application :** IoT applications provide an interface that the users can use to control and monitor various aspects of the IoT system. Applications also allow users to view the system status and view the processed data. Based on how and where these components are deployed, you can have various levels of IoT deployments. Let's learn about them in detail.

## 1.15.1 IoT Deployment Level 1

A level-1 IoT system has a single IoT node (or device) that performs sensing and actuation, stores data, performs analysis and hosts the application. Level-1 IoT systems are suitable for modelling low-cost and low-complexity solutions where the data involved is not big and the analysis requirements are not computationally intensive. As the data involved is not big and the analysis is not computationally intensive, you do not require cloud-based resources for data storage or running analytics. The locally deployed node is sufficient for both storage and processing. The Fig. 1.15.2 shows a high-level diagram for level-1 IoT system.

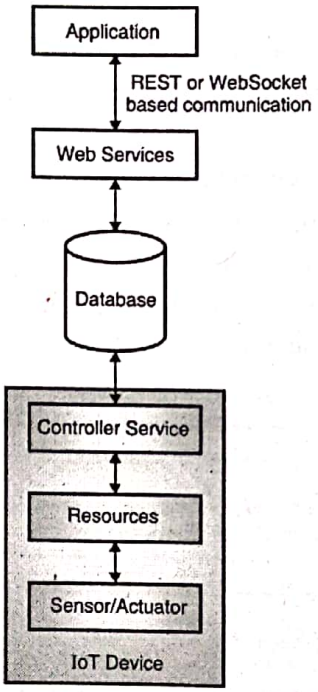



**Fig. 1.15.2 : High-level diagram for level-1 IoT system**

A further simplified diagram of level-1 IoT system is as shown in Fig. 1.15.3.

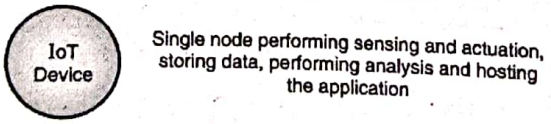



**Fig. 1.15.3 : Simplified diagram of level-1 IoT system**

Basically, a single IoT node is deployed locally, and it performs everything.

Assume that a home automation system, that allows controlling the electronic appliances and lights remotely via an IoT application, is deployed as a level-1 IoT system. In such a deployment, the status information of each appliance could be maintained in the local database. A user could use the IoT application, that could be connected over the Internet, and could control the appliances remotely.

The web service could provide the status of the appliances by reading the database and providing that information to the user via the application. The user could set the appliance status via the application as desired. The web service, on receiving the input from the user via the application, updates the database with the desired status of the appliances.

The controller service interacts with the database continuously and ensures that the appliances are accordingly switched on and off. Reading the updated information from the database, the controller service appropriately actuates the switches of the respective appliances to either turn them on or off as desired by the user.

### 1.15.2 IoT Deployment Level 2

A level-2 IoT system has a single node that performs sensing and actuation and local analysis. Data is stored in the cloud and application is usually cloud-based as well. Level-2 IoT systems are suitable for solutions where the data involved is big, however, the primary analysis requirement is not computationally intensive and hence can be done locally.

The Fig. 1.15.4 shows a high-level diagram for level-2 IoT system.

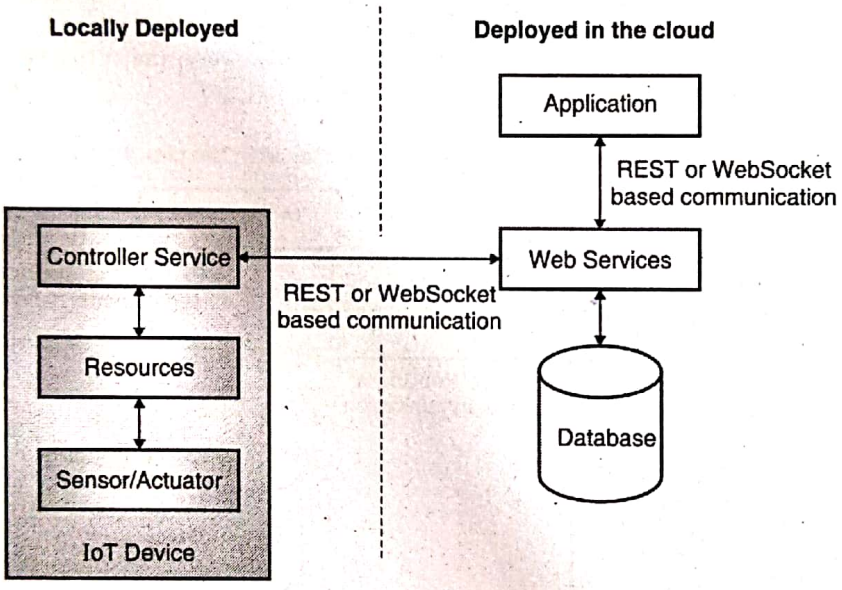



**Fig. 1.15.4 : High-level diagram for level-2 IoT system.**

A further simplified diagram of level-2 IoT system is as shown in Fig. 1.15.5.

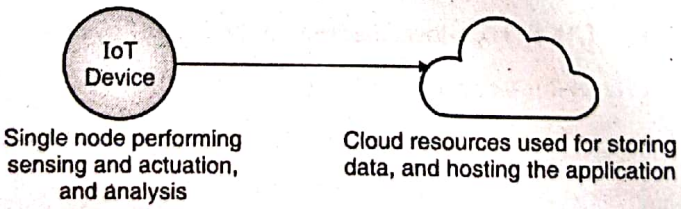



**Fig. 1.15.5 : Simplified diagram of level-2 IoT system**

Assume that a smart irrigation system, that regulates the irrigation system based on the moisture level, is deployed as a level-2 IoT system. A single node could monitor the moisture level continuously and update the moisture level data in the database hosted in the cloud.

The area of the field could be quite large and may be generating around 1000s of moisture level data points per second. The controller service continuously monitors the moisture level to ensure that the moisture levels are as desired. It also interacts with the web service to update the database with the moisture level data.

If there is a drop in the moisture level from the desired level, it could actuate the irrigation valves and turn them on to water the soil for a given amount of time such that to reach the desired moisture level. The application, that is hosted in the cloud, could be used to monitor the activity of the smart irrigation system and also the moisture levels of the soil at various points in the field. The node could further analyse the moisture level data and irrigation schedule (when it is turned on and off) to find the patterns in the collected moisture level data. Based on various environmental conditions, you would be able to predict the demand for water and could ensure that you have adequate supply of it.

### 1.15.3 IoT Deployment Level 3

A level-3 IoT system has a single node. Data is stored and analysed in the cloud and application is cloud-based as well.Level-3 IoT systems are suitable for solutions where the data involved is big and the analysis requirements are computationally intensive. The single node (due to limited storage and processing power) is not sufficient to either store the data or perform analysis on it. In such a scenario, cloud-based resources are used that offer unlimited storage and massive processing power.

The Fig. 1.15.6 shows a high-level diagram for level-3 IoT system. It looks very similar to level-2 IoT system except one change that the analysis is performed in the cloud.

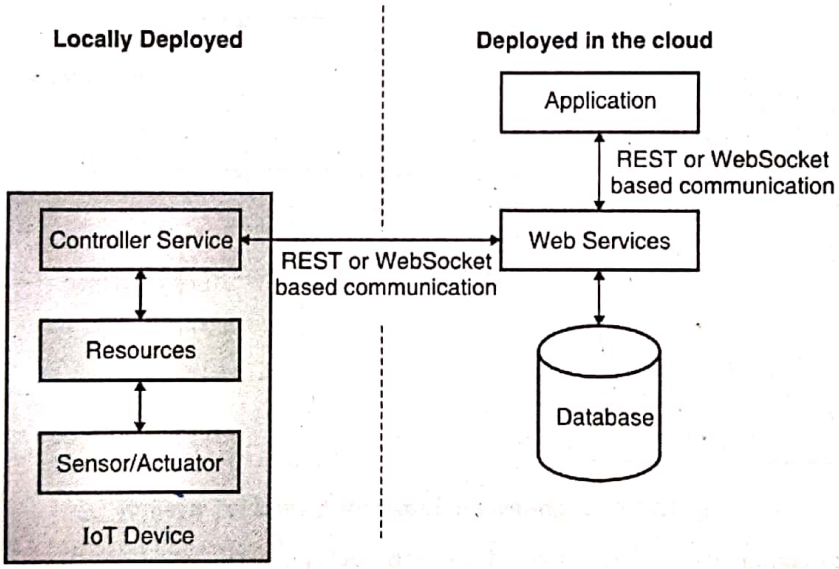



**Fig. 1.15.6 : High-level diagram for level-3 IoT system**

A further simplified diagram of level-3 IoT system is as shown in Fig. 1.15.7.

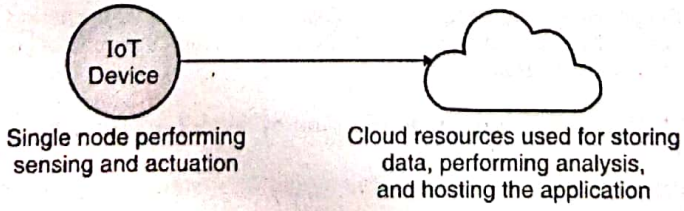



**Fig. 1.15.7 : Simplified diagram of level-3 IoT system**

Assume that a package tracking device, that is attached to the package and tracks the package movement, is deployed as a level-3 IoT system. A single node (attached to the package) could send the package location details (GPS information) to the cloud in real-time using a WebSocket-based service. The location details are saved in the database hosted in the cloud.

You can visualise the package movement details on a global map view in real-time on the tracking application. The location data is continuously analysed in the cloud and anytime the package is detour (deviates from the desired route), the application could raise an alert and could inform the right authorities to take the required actions.

### 1.15.4 IoT Deployment Level 4

A level-4 IoT system has multiple nodes that perform local analysis in a cluster (together they have sufficient resources to perform the analysis locally instead of cloud). Data is stored in the cloud and application is cloud-based as well. A level-4 IoT system contains local and cloud-based observer nodes which can subscribe and receive information collected in the cloud from multiple IoT devices. Level-4 IoT systems are suitable for solutions where multiple nodes are required, the data involved is big and the analysis requirements are computationally intensive. Note here that you could also, optionally, run analytics in the cloud instead of running it locally. The Fig. 1.15.8 shows is a high-level diagram for level-4 IoT system.

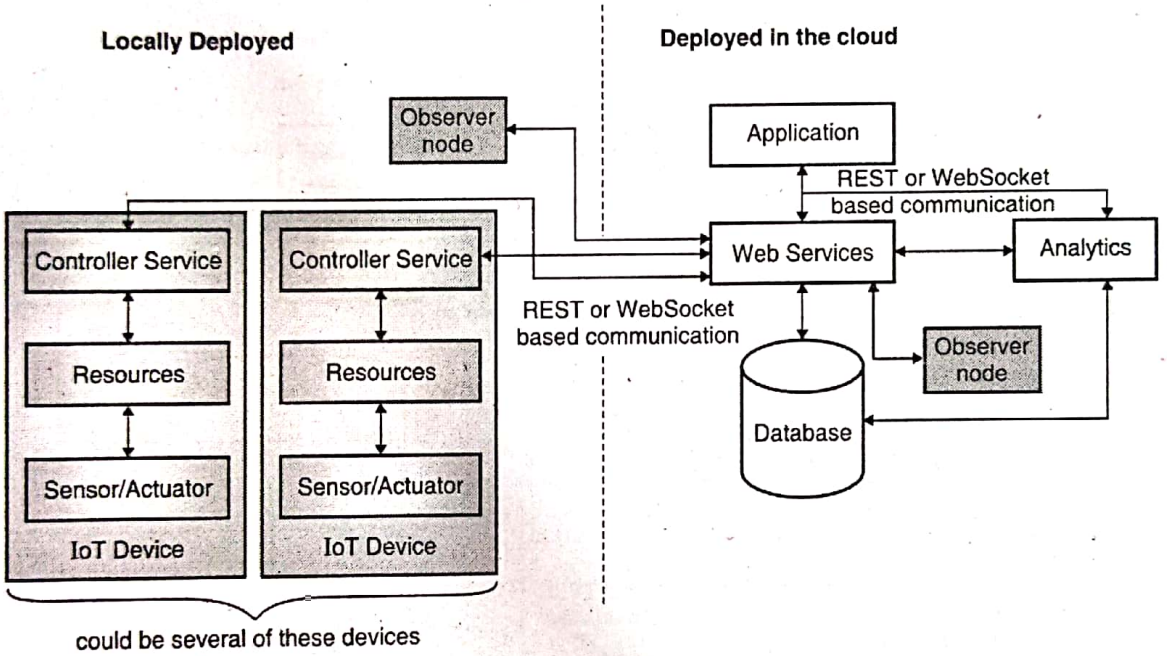



**Fig. 1.15.8 : High-level diagram for level-4 IoT system**

A further simplified diagram of level-4 IoT system is as shown in Fig. 1.15.9.

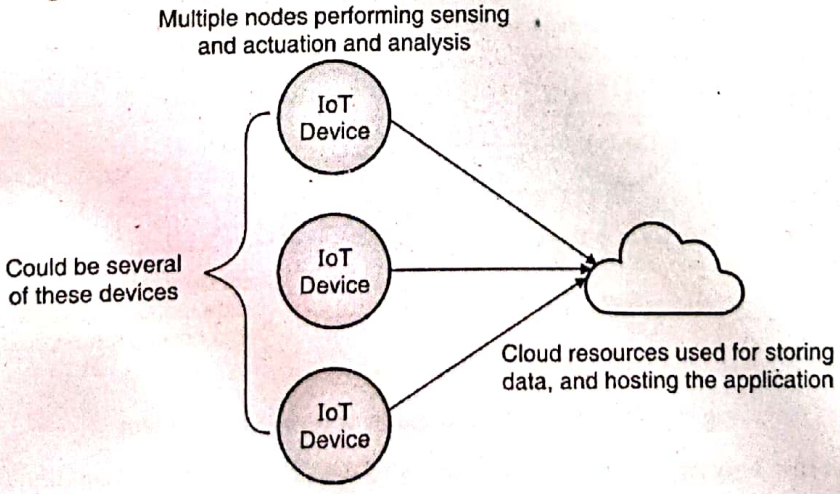



**Fig. 1.15.9 : Simplified diagram of level-4 IoT system**

Assume that a noise monitoring system, having multiple nodes with sound sensors placed in various locations to collect noise level and pattern, is deployed as a level-4 IoT system. Each node is independent and collects the noise level data and sends it to the cloud via the controller service. Noise level data could be stored in the database hosted in cloud. The analysis of noise level could be done either locally or in the cloud. The application could raise an alert when the noise level goes high. You can also visualise noise level at different times to identify patterns.

### 1.15.5 IoT Deployment Level 5

A level-5 IoT system has multiple end nodes and one coordinator node. The end nodes perform sensing and actuation. The coordinator node collects the data from the end nodes and sends it to the cloud. Data is stored and analysed in the cloud and application is cloud-based as well. Level-5 IoT systems are suitable for solutions based on wireless sensor networks, in which the data involved is big and the analysis requirements are computationally intensive.

The Fig. 1.15.10 shows a high-level diagram for level-5 IoT system.

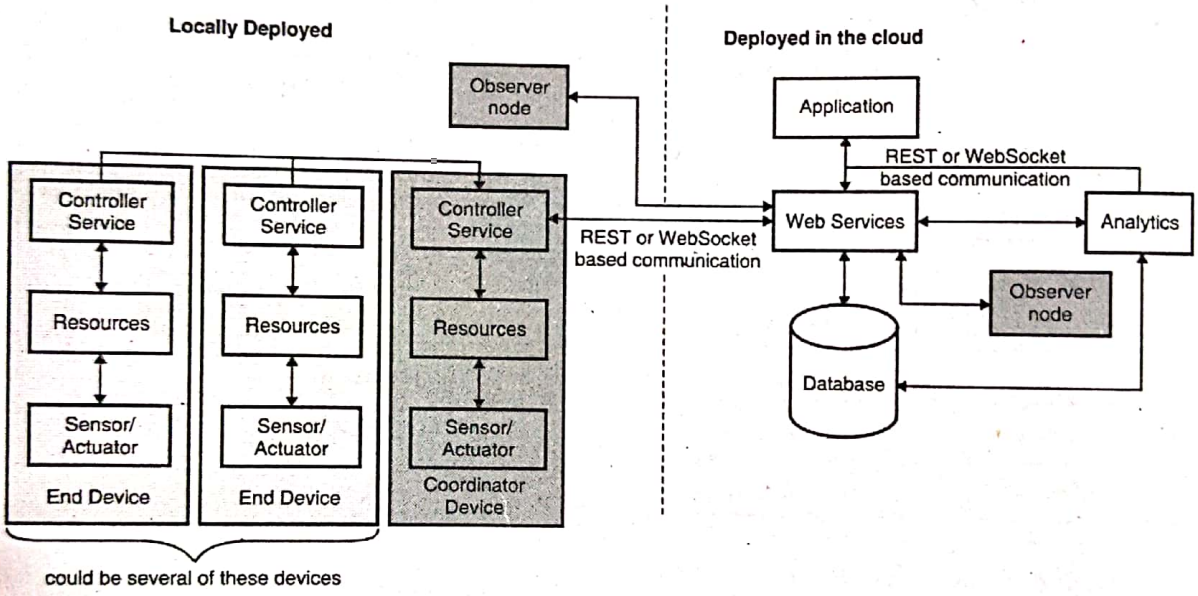



**Fig. 1.15.10 : High-level diagram for level-5 IoT system**

A further simplified diagram of level-5 IoT system is as shown in Fig. 1.15.11.

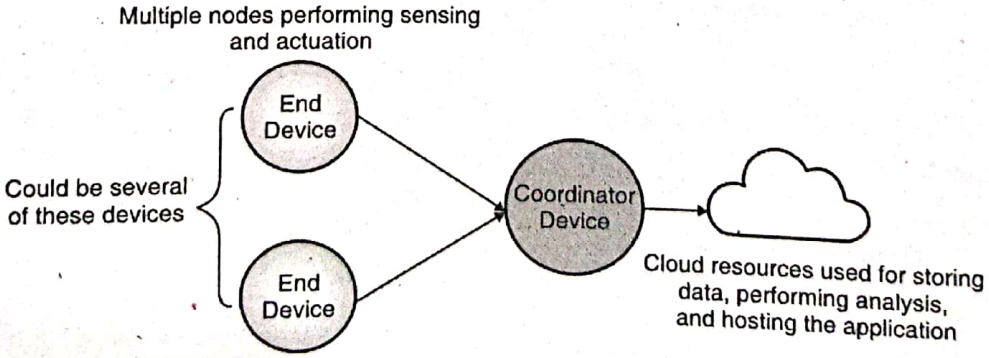



**Fig. 1.15.11 : Simplified diagram of level-5 IoT system**

Assume that a forest fire detection system, having sensors to detect temperature, humidity, and $CO_2$ levels from the surroundings, is deployed as a level-5 IoT system. Multiple end nodes are placed at different locations for monitoring temperature, humidity, and $CO_2$ levels throughout the forest.

The data sensed and collected by these end nodes is aggregated by the coordinator node and is sent to the cloud by the controller service running on the coordinator node.

The coordinator node acts as a gateway that provides internet connectivity to the level-5 IoT system. The advantage of this kind of a gateway-based setup is that the end nodes may not require internet connectivity of their own. End nodes could just have the connectivity up to the gateway (coordinator node) and only the coordinator node needs to be connected to the Internet.

The data is stored in a database hosted in the cloud. The data is continuously analysed by the analytics service running in the cloud. Based on the collected data, the analytics service could find patterns that predict the possibility of an occurrence of forest fire (depending upon temperature, humidity, and $CO_2$ levels) and could proactively alert the right set of people for taking appropriate actions. The nodes and their collected data could be visualised and monitored via the cloud-based application.

## 1.15.6 IoT Deployment Level 6

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

The Fig. 1.15.12 is a high-level diagram for level-6 IoT system.

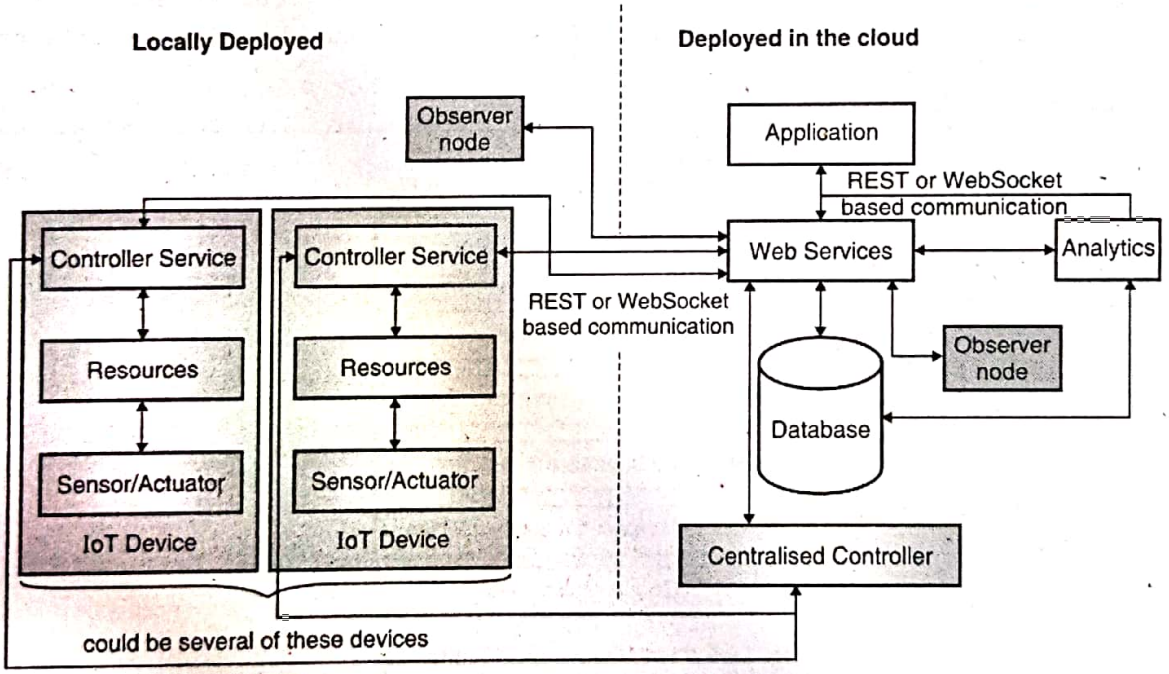



**Fig. 1.15.12 : High-level diagram for level-6 IoT system**

A further simplified diagram of level-6 IoT system is as shown in Fig. 1.15.13.

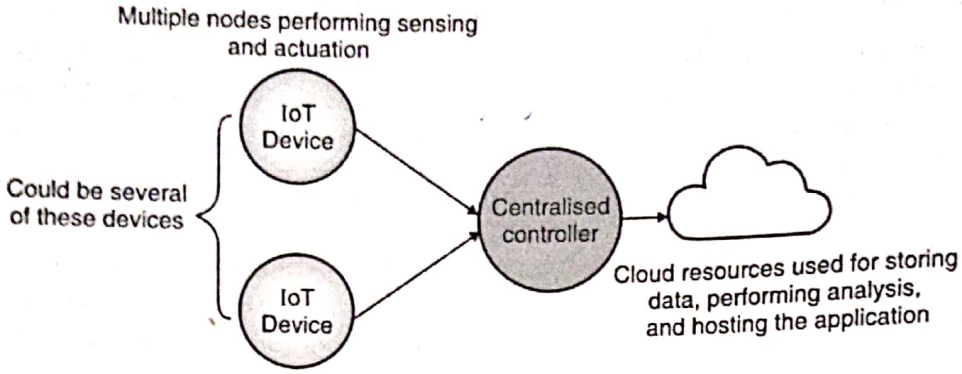



**Fig. 1.15.13 : Simplified diagram of level-6 IoT system**

Assume that a weather monitoring system, having sensors to detect temperature, humidity, and pressure levels from the surroundings, is deployed as a level-6 IoT system. Multiple nodes are placed at different locations for monitoring temperature, humidity, and pressure levels throughout the monitored area. The data sensed and collected by these nodes is sent to the cloud by the controller service running on the nodes.

The data is stored in a database hosted in the cloud. The data is continuously analysed by the analytics service running in the cloud. Based on the collected data, the analytics service could find patterns that predicts weather (depending upon temperature, humidity, and pressure levels) and could proactively alert the right set of people for taking appropriate actions. The nodes and their collected data could be visualised and monitored via the cloud-based application.

## 1.16 IoT Issues and Challenges

Some of the major challenges to consider in IoT systems are as shown in Fig. 1.16.1.

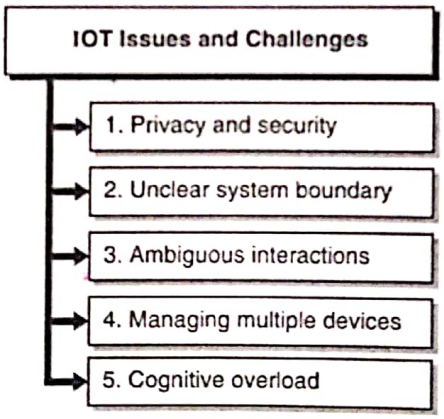



**Fig. 1.16.1 : IoT Issues and Challenges**

1. **Privacy and Security :** IoT systems can pose privacy and security challenges. Such systems collect and work with a lot of personal data and ensuring that the personal data is not misused is a big responsibility on the data collecting organisation. To improve the services provided by the IoT systems, the personal data should be anonymised and kept secured. Additionally, the device identity and its lifecycle from onboarding to disposal must be securely maintained. A rogue (malicious) device should not be able to register itself on the secure IoT network and pilferage (steal) or corrupt any data.
2. **Unclear system boundary :** IoT systems often work with wireless networks. Unlike wired networks, wireless networks are hard to keep within a physical boundary. The data transferred over a wireless network can be captured from outside the physical boundaries leading to information leakage.

Also, it is possible to control the IoT system externally by forcing them to connect to a malicious wireless network. There should be adequate measures taken to protect the IoT devices from takeover.

3. **Ambiguous interactions :** IoT systems often work through sensors. For example, turning on the light when you enter the room and turning it off when you leave the room. A user does not have a direct control over the system. Sometimes, these sensors may falsely detect a condition. For example, if someone else enters the room while you are still in the room, and then leaves after talking to you, the sensor may "believe" that you have left as well and turn off the lights. You then may have to make a sound or go out and come in again. Such ambiguous and unclear interactions might be annoying and may not deliver the optimum user experience.
4. **Managing multiple devices :** A building might have several IoT systems interconnected with each other. Managing, operating, and maintaining them can pose a challenge. You often require a centralised system for administration. Also, these devices are usually placed far from common reach.

   So, if any IoT device requires any physical handling, it might be tough to quickly identify it and to reach it for any maintenance activity.
5. **Cognitive overload :** The user may feel overwhelmed with so many automated devices around. What works how, which interactions and gestures to make, what to say, how to control – all this could put mental pressure in the day to day life of the user.

## 1.17 IoT and M2M

As you understand, IoT systems have various functional blocks. There are several protocols, standards and mechanisms that enable these IoT systems to interact with one another and also with the outside world and make IoT systems useful and effective. These technologies existed way before IoT became a domain in itself. IoT has reused several of these concepts, technologies and mechanisms and have extended them further in various ways to evolve into a domain of its own.

### 1.17.1 Introduction to M2M

Have you ever used your phone to cast its screen or video on a TV screen? Have you streamed music from your phone to playback on a Bluetooth speaker or headphone? If yes, you have experienced some form of M2M already!

Machine-to-Machine or M2M can be rightly articulated as the Internet of devices. Several devices communicate and interact with each other coherently to carry out useful tasks.

Hence,

✍ ***Definition :*** *M2M refers to technologies, standards and protocols that enable the machines to communicate and interact with each other and carry out useful tasks.*

Traditionally, the devices required to be connected to a central hub for any communication but nowadays they could also be connected directly as the end point devices, if desired. M2M can use either the wired medium or the wireless medium for communication. Some of the common examples of M2M communication technologies are cellular networks, ethernet, Wi-Fi, Bluetooth, and Infrared.

In the IoT world, M2M concept is generally extended to refer to the collection of sensor data from various devices and have it processed to carry out higher-level tasks such as prediction, automation and alerting. Note here that IoT uses M2M technologies and not the other way round.

## 1.17.1(A) Applications of M2M

As I said, M2M existed way before IoT became a domain of its own. Some of the common applications of M2M are as following.

1. **Robotics :** M2M is used in robotics to place inventory in warehouses and auto fill shipment orders. It is also used in various plants and assembly lines in to automate manufacturing related tasks.
2. **Logistics and fleet management :** You could use M2M for fleet management where you could track movement of vehicles and objects and efficiently manage them. For example you could collect information such as location, timings, traffic jams, and in-route environmental conditions and send this information to an application that can track the vehicle and deliveries.
3. **Utility :** You can use M2M in monitoring smart utility meters for electricity, water, gas or anything else. A smart meter is an electronic device that records consumption of electricity, gas or water and communicates that information for monitoring and billing. Smart meters send meter readings to the utility company automatically. They also come with in-home displays, which give users real-time feedback on their energy or water usage and what it is costing.
4. **Vending machines :** Instead of having someone to go and check vending machines several times in a day for refilling and maintenance, M2M can be used for providing the current operational status and stock of items in the vending machines. You are then required to send someone for refilling or maintenance only when it is required instead of sending someone periodically.

## 1.17.1(B) General High-Level Architecture of M2M (Defined by ETSI)

The Fig. 1.17.1 illustrates the M2M architecture defined by ETSI (The European Telecommunications Standards Institute).

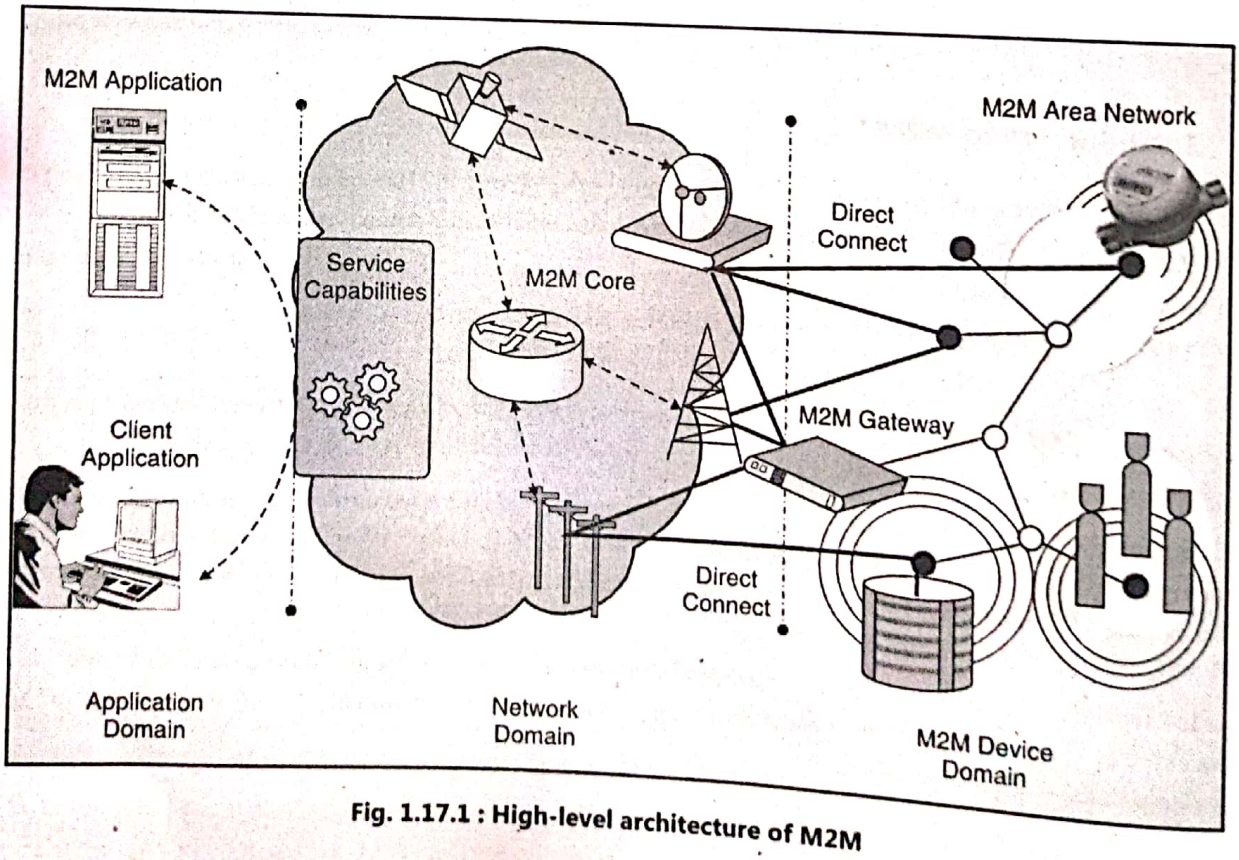



**Fig. 1.17.1 : High-level architecture of M2M**

It could be simplified in Fig. 1.17.2.

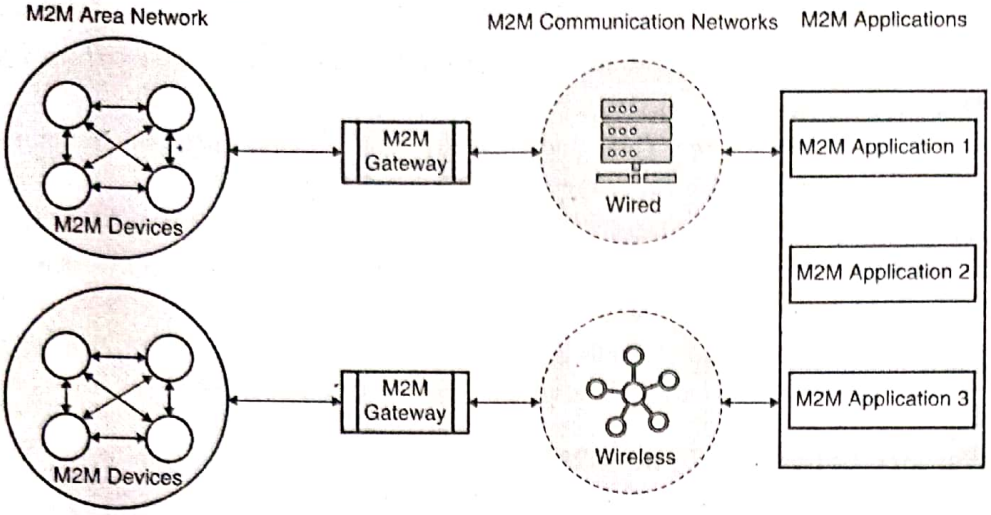



**Fig. 1.17.2**

1. **M2M Device :** This could be any device capable of replying to request for data or capable of transmitting data of its own.
2. **M2M Area Network (Device Domain) :** It provides connectivity between M2M Devices and M2M Gateways. For example, personal area network.
3. **M2M Gateway :** It provides capabilities for interconnecting M2M device to the communication network.
4. **M2M Communication Networks (Network Domain) :** These provide communications between the M2M Gateway(s) and M2M application(s). For example, LTE, WiMAX, and WLAN.
5. **M2M Applications :** This is the middleware layer where data goes through various application services and is used by the specific business-processing programs.

## 1.17.1(C) High-level Architecture of M2M for IoT

The Fig. 1.17.3 shows the high-level architecture of how M2M technologies are used in IoT.

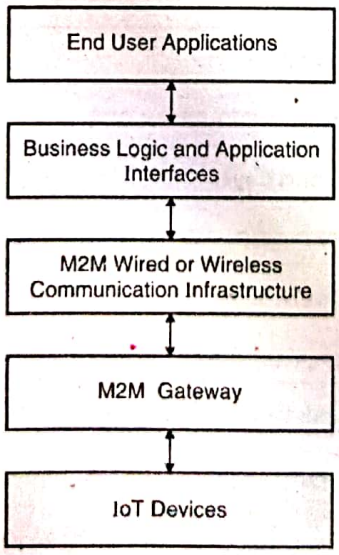



**Fig. 1.17.3 : High-level architecture of M2M in IoT**

1. **IoT Devices :** These could be any IoT device that you wish to interact with. It could be TV, refrigerator, car or anything else.
2. **M2M Gateway :** These are devices that run communication protocols to collect output data from the IoT devices or provide inputs.
3. **M2M Communication Infrastructure :** This could use either wired or wireless communication medium. There are several protocols and standards available such as ethernet, Wi-Fi, and Bluetooth.
4. **Business Logic and Application Interfaces :** This layer is also called as M2M middleware. You can process the collected data at this layer and expose it to end user applications for consumption. It provides APIs using which you can develop applications for interacting with the IoT devices.
5. **End User Applications :** End user applications is what you use to interact with the IoT devices. You can run these applications on your smartphone, tablet, PC, kiosk, vending machine or anywhere else as appropriate.

## 1.17.1(D) Difference between IoT and M2M

Sometimes, it might feel like M2M and IoT are referring to same thing. But, there are quite a few differences between them. The Table 1.17.1 provides a comparison between M2M and IoT.

**Table 1.17.1 : Comparison between M2M and IoT**

| Comparison Attribute | M2M | IoT |
|---|---|---|
| Technology for | Machine-to-Machine communication | Connected things (sensors and actuators) |
| Applications used are | Vertical applications | Horizontal applications |
| IP protocol | Not used | Used |
| Logic embedded in | Hardware | Hardware and software |
| Interoperability | Low | High |
| Scalability | Low | High |
| Internet connectivity | Rare | Often |

The Fig. 1.17.4 highlights the communication protocols, that you read about in Chapter 1, as used for M2M and IoT communication respectively.

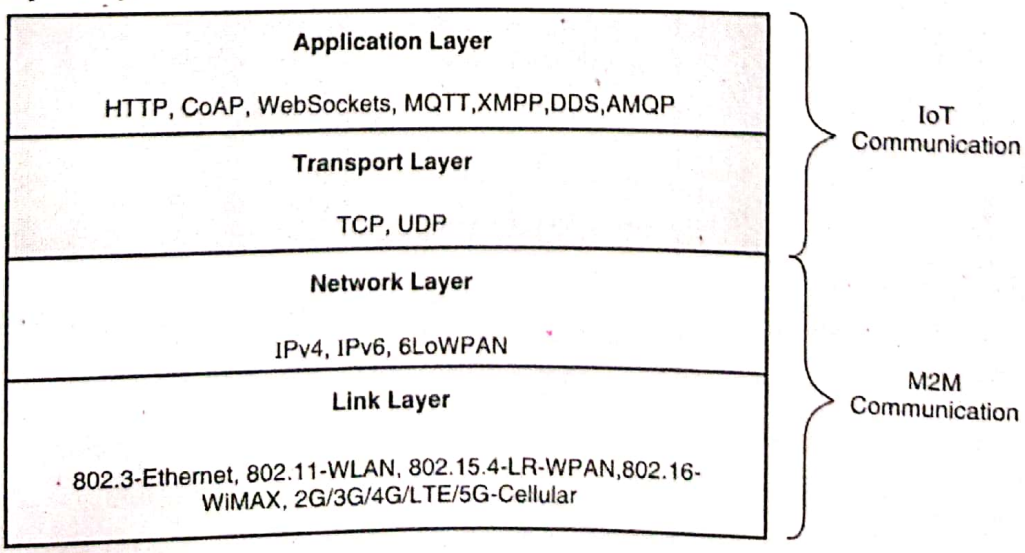



**Fig. 1.17.4 : Communication protocols**

## 1.18 Web of Things (WoT)

Technically speaking, the term Internet actually refers to an electronic communications network that connects several computer systems around the world. There could be a wide range of networking protocols, such as FTP, SSH, SMTP, SNMP, HTTP, that could be used for building such a massive network called the Internet.

However, as you would agree, when you talk about the Internet, you usually mean something that you do with browsers or connected apps that work over HTTP protocol (WWW – World Wide Web). Out of the several networking protocols, HTTP has become the major protocol for the Internet.

The reason the Internet picked up and became so common was perhaps because anyone could build applications to use the Internet in the standard way. Applications are what users use and developers building these applications find it hard to work with 100s of protocols instead of the standard and popular ones.

✍ ***Definition :*** *The Web of Things (WoT) is an attempt to not stop at the Internet level with respect to IoT but build hooks to connect IoT systems to a generic web application architecture.*

In the IoT, hundreds of incompatible protocols co-exist today. This makes the integration of data and services from various devices extremely complex and costly. In the Web of Things, any device can be accessed using standard Web protocols. Connecting heterogeneous devices to the Web makes the integration across systems and applications much simpler. Any device could be easily integrated and consumed by any application, regardless of the networking protocols used by those devices. This is exactly what the Web of Things enables you to do.

The Fig. 1.18.1 highlights what WoT is. Despite the heterogeneous networking protocols to control and operate IoT devices, a user via a web application could potentially connect, manage, and operate (or make use of) these devices. The standard web protocols and architecture such as HTTP, JavaScript, REST, JSON, WebSocket are typically used to build WoT.

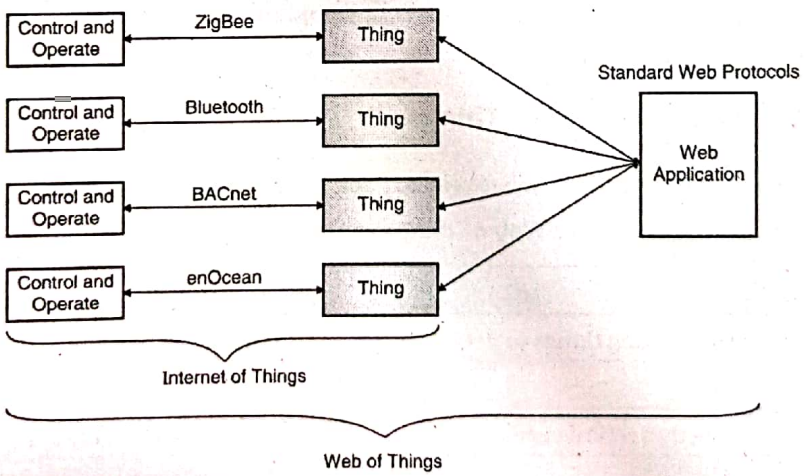



**Fig. 1.18.1 : Web of things**

As per the OSI model, WoT could be viewed as the Fig. 1.18.2.

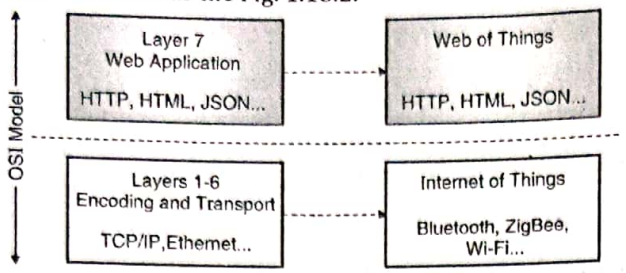



**Fig. 1.18.2 : Web of things OSI**

## 1.18.1 Two Pillars of the Web

At a high-level, the web today can be considered as being supported by two foundational pillars – standard protocols and applications based off these standard protocols.

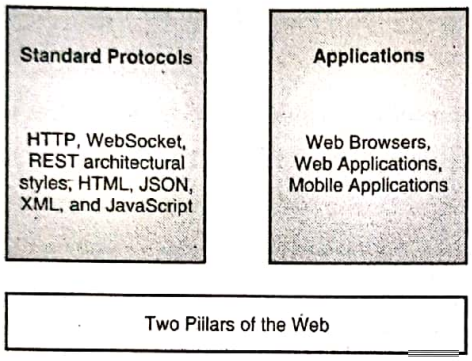



**Fig. 1.18.3 : Two pillars of the web**

Some of the most common protocols and standards in use are HTTP, WebSocket, REST architectural styles, HTML, JSON, XML, and JavaScript. Some of the most common applications are web browsers, web-based applications, and mobile apps. Standard protocols help to work with the web resources and applications provide a way to consume or interact with those resources through these standard protocols.

## 1.18.2 Web of Things versus Internet of Things

The Table 1.18.1 provides a quick comparison between IoT and WoT.

**Table 1.18.1 : Web of things versus internet of things**

| Comparison Attribute | IoT | WoT |
|---|---|---|
| Scope | Connecting things to networks | Connecting thing to web applications |
| OSI Layers | 1-6 | 7 |
| Protocol standards | Mostly vertical specific | Mostly horizontally applicable |
| Protocols Used | Networking level protocols | Application level protocols and architecture |
| Control | Low (at a per thing level) | High (multiple things at once) |
| Vendor lock-in | Yes | No |
| Developer friendly | No | Yes |

## Review Questions

Here are a few review questions to help you gauge your understanding of this chapter. Try to attempt these questions and ensure that you can recall the points mentioned in the chapter.

### [A] Introduction to Internet of Things (IoT)

| | | |
|---|---|---|
| **Q. 1** | Write a short note on IoT. | **[4 Marks]** |
| **Q. 2** | Describe the major characteristics of IoT. | **[6 Marks]** |
| **Q. 3** | Write a short note on IoT vision focusing on how it could impact human-beings. | **[4 Marks]** |
| **Q. 4** | Write a short note on IoT vision focusing on how it could impact homes. | **[4 Marks]** |
| **Q. 5** | Write a short note on IoT vision focusing on how it could impact retail. | **[4 Marks]** |
| **Q. 6** | Write a short note on IoT vision focusing on how it could impact factories. | **[4 Marks]** |
| **Q. 7** | Write a short note on IoT vision focusing on how it could impact vehicles. | **[4 Marks]** |
| **Q. 8** | Write a short note on IoT vision focusing on how it could impact cities. | **[4 Marks]** |
| **Q. 9** | A precise definition for IoT is still evolving. Write a few IoT definitions that you hear commonly. | **[4 Marks]** |
| **Q. 10** | List a few IoT standard frameworks of your choice. | **[4 Marks]** |
| **Q. 11** | Explain the types of capabilities in IoT devices. | **[6 Marks]** |
| **Q. 12** | Write a short note on transducer capabilities in IoT devices. | **[4 Marks]** |
| **Q. 13** | Write a short note on interface capabilities in IoT devices. | **[4 Marks]** |
| **Q. 14** | With a block diagram, explain technical building blocks of IoT. | **[6 Marks]** |
| **Q. 15** | With a block diagram, explain high-level architecture of IoT. | **[6 Marks]** |
| **Q. 16** | Write a short note on IoT gateway. | **[4 Marks]** |

### [B] Physical Design of IoT

| | | |
|---|---|---|
| **Q. 17** | List the IoT protocols at various OSI layers. | **[4 Marks]** |
| **Q. 18** | Explain a few link layer IoT protocols. | **[6 Marks]** |
| **Q. 19** | Write a short note on 802.3 – Ethernet with respect to IoT protocols. | **[4 Marks]** |
| **Q. 20** | Write a short note on 802.11 – WLAN with respect to IoT protocols. | **[4 Marks]** |
| **Q. 21** | Write a short note on 802.15.4 – LR-WPAN with respect to IoT protocols. | **[4 Marks]** |
| **Q. 22** | Write a short note on 802.16 – WiMAX with respect to IoT protocols. | **[4 Marks]** |
| **Q. 23** | Write a short note on 2G / 3G / 4G / LTE / 5G – Cellular with respect to IoT protocols. | **[4 Marks]** |
| **Q. 24** | Write a short note on IPv4 with respect to IoT protocols. | **[4 Marks]** |
| **Q. 25** | Write a short note on IPv6 with respect to IoT protocols. | **[4 Marks]** |
| **Q. 26** | Write a short note on 6LoWPAN with respect to IoT protocols. | **[4 Marks]** |

Q. 27 Write a short note on TCP with respect to IoT protocols. [4 Marks]

Q. 28 Write a short note on UDP with respect to IoT protocols. [4 Marks]

Q. 29 Write a short note on HTTP with respect to IoT protocols. [4 Marks]

Q. 30 Write a short note on CoAP with respect to IoT protocols. [4 Marks]

Q. 31 Write a short note on WebSockets with respect to IoT protocols. [4 Marks]

Q. 32 Write a short note on MQTT with respect to IoT protocols. [4 Marks]

Q. 33 Write a short note on XMPP with respect to IoT protocols. [4 Marks]

Q. 34 Write a short note on DDS with respect to IoT protocols. [4 Marks]

Q. 35 Write a short note on AMQP with respect to IoT protocols. [4 Marks]

**[C] Logical Design of IoT**

Q. 36 With a block diagram, explain the various functional blocks in IoT. [6 Marks]

Q. 37 Classify IoT communication models and briefly explain each. [6 Marks]

Q. 38 With a block diagram, explain request-response communication model in IoT. [6 Marks]

Q. 39 With a block diagram, explain publish-subscribe communication model in IoT. [6 Marks]

Q. 40 With a block diagram, explain push-pull communication model in IoT. [6 Marks]

Q. 41 With a block diagram, explain exclusive pair communication model in IoT. [6 Marks]

Q. 42 Compare various IoT communication models. [6 Marks]

Q. 43 Describe REST-based communication APIs with respect to IoT. [6 Marks]

Q. 44 Write a short note on resources with respect to REST. [4 Marks]

Q. 45 Briefly explain request methods with respect to REST. [4 Marks]

Q. 46 Describe REST architectural constraints. [6 Marks]

Q. 47 With an example, explain layered system with respect to REST. [4 Marks]

Q. 48 Describe WebSocket-based communication APIs with respect to IoT. [6 Marks]

Q. 49 Compare REST-based and WebSocket-based APIs. [6 Marks]

**[D] IoT Enabling Technologies**

Q. 50 Briefly explain IoT enabling technologies. [6 Marks]

Q. 51 Write a short note on embedded systems. [4 Marks]

Q. 52 With a simple block diagram, explain the purpose of embedded systems. [6 Marks]

Q. 53 Joe is arguing that electronic handheld tablet device is an example of embedded system. What do you tell him? [4 Marks]

Q. 54 Compare embedded systems with general purpose computing systems. [6 Marks]

**Q. 55** You can also install a full version of Windows 10 OS, that runs on a laptop, on an embedded system. Comment. **[4 Marks]**

**Q. 56** Describe a few applications of embedded systems. **[6 Marks]**

**Q. 57** What are some of the domains in which embedded systems are commonly used? **[4 Marks]**

**Q. 58** Describe the major characteristics of embedded systems. **[6 Marks]**

**Q. 59** Embedded systems have large form factor. Comment. **[4 Marks]**

**Q. 60** Embedded systems have small form factor. Comment. **[4 Marks]**

**Q. 61** Write a short note on Wireless Sensor Networks (WSNs). **[4 Marks]**

**Q. 62** Write a short note on Big Data Analytics with respect to IoT. **[4 Marks]**

**Q. 63** Describe the characteristics of Big Data. **[6 Marks]**

**Q. 64** Describe the five Vs of Big Data. **[6 Marks]**

**Q. 65** Explain the role of cloud computing in IoT. **[6 Marks]**

**Q. 66** Explain the goals of cloud computing. **[6 Marks]**

### [E] IoT Levels and Deployment Templates

**Q. 67** With a block diagram, explain IoT deployment level 1. Also, give an example of how such a system could work. **[6 Marks]**

**Q. 68** With a block diagram, explain IoT deployment level 2. Also, give an example of how such a system could work. **[6 Marks]**

**Q. 69** With a block diagram, explain IoT deployment level 3. Also, give an example of how such a system could work. **[6 Marks]**

**Q. 70** With a block diagram, explain IoT deployment level 4. Also, give an example of how such a system could work. **[6 Marks]**

**Q. 71** With a block diagram, explain IoT deployment level 5. Also, give an example of how such a system could work. **[6 Marks]**

**Q. 72** With a block diagram, explain IoT deployment level 6. Also, give an example of how such a system could work. **[6 Marks]**

### [F] IoT Issues and Challenges

**Q. 73** Explain the major issues and challenges with IoT. **[6 Marks]**

### [G] IoT and M2M

**Q. 74** Write a short note on M2M. **[4 Marks]**

**Q. 75** Describe a few applications of M2M. **[4 Marks]**

**Q. 76** With a block diagram, explain high-level architecture of M2M. **[6 Marks]**

**Q. 77** Explain high-level architecture of M2M for IoT. **[6 Marks]**

**Q. 78** Compare IoT and M2M. **[6 Marks]**

**Q. 79** List the protocols used in M2M and IoT. **[4 Marks]**

**[H] Web of Things**

**Q. 80** Using a suitable block diagram, explain Web of Things. **[6 Marks]**

**Q. 81** Write a short note on Web of Things. **[4 Marks]**

**Q. 82** Write a short note on two pillars of the web. **[4 Marks]**

**Q. 83** Compare Web of Things with Internet of Things. **[6 Marks]**

❑❑❑

## LEC-9: SQL in 1-Video

1. **SQL**: Structured Query Language, used to access and manipulate data.
2. SQL used **CRUD** operations to communicate with DB.
    1. **CREATE** - execute INSERT statements to insert new tuple into the relation.
    2. **READ** - Read data already in the relations.
    3. **UPDATE** - Modify already inserted data in the relation.
    4. **DELETE** - Delete specific data point/tuple/row or multiple rows.
3. **SQL is not DB, is a query language.**
4. What is **RDBMS**? (Relational Database Management System)
    1. Software that enable us to implement designed relational model.
    2. e.g., MySQL, MS SQL, Oracle, IBM etc.
    3. Table/Relation is the simplest form of data storage object in R-DB.
    4. **MySQL** is open-source RDBMS, and it uses SQL for all CRUD operations
5. **MySQL** used client-server model, where client is CLI or frontend that used services provided by MySQL server.
6. **Difference between SQL and MySQL**
    1. SQL is Structured Query language used to perform CRUD operations in R-DB, while MySQL is a RDBMS used to store, manage and administrate DB (provided by itself) using SQL.

**SQL DATA TYPES** (Ref: https://www.w3schools.com/sql/sql_datatypes.asp)

1. In SQL DB, data is stored in the form of tables.
2. Data can be of different types, like INT, CHAR etc.

| DATATYPE | Description |
|---|---|
| CHAR | string(0-255), string with size = (0, 255], e.g., CHAR(251) |
| VARCHAR | string(0-255) |
| TINYTEXT | String(0-255) |
| TEXT | string(0-65535) |
| BLOB | string(0-65535) |
| MEDIUMTEXT | string(0-16777215) |
| MEDIUMBLOB | string(0-16777215) |
| LONGTEXT | string(0-4294967295) |
| LONGBLOB | string(0-4294967295) |
| TINYINT | integer(-128 to 127) |
| SMALLINT | integer(-32768 to 32767) |
| MEDIUMINT | integer(-8388608 to 8388607) |
| INT | integer(-2147483648 to 2147483647) |
| BIGINT | integer (-9223372036854775808 to 9223372036854775807) |
| FLOAT | Decimal with precision to 23 digits |
| DOUBLE | Decimal with 24 to 53 digits |

| DATATYPE | Description |
|---|---|
| DECIMAL | Double stored as string |
| DATE | YYYY-MM-DD |
| DATETIME | YYYY-MM-DD HH:MM:SS |
| TIMESTAMP | YYYYMMDDHHMMSS |
| TIME | HH:MM:SS |
| ENUM | One of the preset values |
| SET | One or many of the preset values |
| BOOLEAN | 0/1 |
| BIT | e.g., BIT(n), n upto 64, store values in bits. |

3. Size: TINY < SMALL < MEDIUM < INT < BIGINT.
4. **Variable length Data types** e.g., VARCHAR, are better to use as they occupy space equal to the actual data size.
5. Values can also be unsigned e.g., INT UNSIGNED.
6. **Types of SQL commands:**
    1. **DDL** (data definition language): defining relation schema.
        1. **CREATE**: create table, DB, view.
        2. **ALTER TABLE**: modification in table structure. e.g, change column datatype or add/remove columns.
        3. **DROP**: delete table, DB, view.
        4. **TRUNCATE**: remove all the tuples from the table.
        5. **RENAME**: rename DB name, table name, column name etc.
    2. **DRL/DQL** (data retrieval language / data query language): retrieve data from the tables.
        1. **SELECT**
    3. **DML** (data modification language): use to perform modifications in the DB
        1. **INSERT**: insert data into a relation
        2. **UPDATE**: update relation data.
        3. **DELETE**: delete row(s) from the relation.
    4. **DCL** (Data Control language): grant or revoke authorities from user.
        1. **GRANT**: access privileges to the DB
        2. **REVOKE**: revoke user access privileges.
    5. **TCL** (Transaction control language): to manage transactions done in the DB
        1. **START TRANSACTION**: begin a transaction
        2. **COMMIT**: apply all the changes and end transaction
        3. **ROLLBACK**: discard changes and end transaction
        4. **SAVEPOINT**: checkout within the group of transactions in which to rollback.

**MANAGING DB (DDL)**

1. **Creation of DB**
    1. **CREATE DATABASE IF NOT EXISTS db-name;**
    2. **USE** db-name; //need to execute to choose on which DB CREATE TABLE etc commands will be executed. //make switching between DBs possible.
    3. **DROP** DATABASE IF EXISTS db-name; //dropping database.
    4. **SHOW** DATABASES; //list all the DBs in the server.
    5. **SHOW** TABLES; //list tables in the selected DB.

**DATA RETRIEVAL LANGUAGE (DRL)**

1. Syntax: SELECT <set of column names> FROM <table_name>;
2. Order of execution from RIGHT to LEFT.
3. Q. Can we use SELECT keyword without using FROM clause?
    1. Yes, using DUAL Tables.
    2. Dual tables are dummy tables created by MySQL, help users to do certain obvious actions without referring to user defined tables.
    3. e.g., SELECT 55 + 11;
       SELECT now();
       SELECT ucase(); etc.
4. **WHERE**
    1. Reduce rows based on given conditions.
    2. E.g., SELECT * FROM customer WHERE age > 18;
5. **BETWEEN**
    1. SELECT * FROM customer WHERE age between 0 **AND** 100;
    2. In the above e.g., 0 and 100 are inclusive.
6. **IN**
    1. Reduces **OR** conditions;
    2. e.g., SELECT * FROM officers WHERE officer_name IN ('Lakshay', 'Maharana Pratap', 'Deepika');
7. **AND/OR/NOT**
    1. AND: WHERE cond1 AND cond2
    2. OR: WHERE cond1 OR cond2
    3. NOT: WHERE col_name NOT IN (1,2,3,4);
8. **IS NULL**
    1. e.g., SELECT * FROM customer WHERE prime_status is NULL;
9. **Pattern Searching / Wildcard ('%', '_')**
    1. '%', any number of character from 0 to n. Similar to '*' asterisk in regex.
    2. '_', only one character.
    3. SELECT * FROM customer WHERE name **LIKE** '%p_';
10. **ORDER BY**
    1. Sorting the data retrieved using **WHERE** clause.
    2. ORDER BY <column-name> DESC;
    3. DESC = Descending and ASC = Ascending
    4. e.g., SELECT * FROM customer ORDER BY name DESC;
11. **GROUP BY**
    1. GROUP BY Clause is used to collect data from multiple records and group the result by one or more column. It is generally used in a SELECT statement.
    2. Groups into category based on column given.
    3. SELECT c1, c2, c3 FROM sample_table WHERE cond GROUP BY c1, c2, c3.
    4. All the column names mentioned after SELECT statement shall be repeated in GROUP BY, in order to successfully execute the query.
    5. Used with aggregation functions to perform various actions.
        1. COUNT()
        2. SUM()
        3. AVG()
        4. MIN()
        5. MAX()
12. **DISTINCT**
    1. Find distinct values in the table.
    2. SELECT DISTINCT(col_name) FROM table_name;
    3. GROUP BY can also be used for the same
        1. "Select col_name from table GROUP BY col_name;" same output as above DISTINCT query.

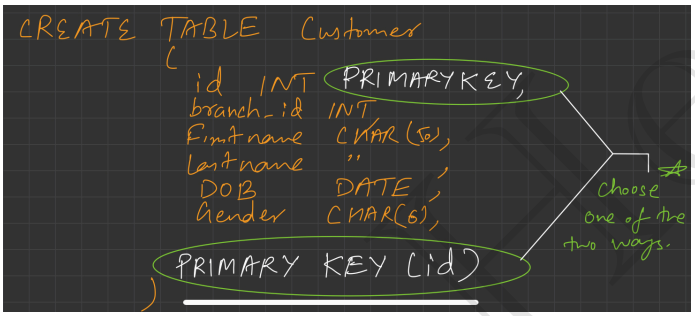
2. SQL is smart enough to realise that if you are using GROUP BY and not using any aggregation function, then you mean "DISTINCT".

**13. GROUP BY HAVING**

1. Out of the categories made by GROUP BY, we would like to know only particular thing (cond).
2. Similar to WHERE.
3. Select COUNT(cust_id), country from customer GROUP BY country HAVING COUNT(cust_id) > 50;
4. WHERE vs HAVING
    1. Both have same function of filtering the row base on certain conditions.
    2. WHERE clause is used to filter the rows from the table based on specified condition
    3. HAVING clause is used to filter the rows from the groups based on the specified condition.
    4. HAVING is used after GROUP BY while WHERE is used before GROUP BY clause.
    5. If you are using HAVING, GROUP BY is necessary.
    6. WHERE can be used with SELECT, UPDATE & DELETE keywords while GROUP BY used with SELECT.

**CONSTRAINTS (DDL)**

1. **Primary Key**




    1. PK is not null, unique and only one per table.
2. **Foreign Key**
    1. FK refers to PK of other table.
    2. Each relation can having any number of FK.
    3. CREATE TABLE ORDER (
    id INT PRIMARY KEY,
    delivery_date DATE,
    order_placed_date DATE,
    cust_id INT,
    FOREIGN KEY (cust_id) REFERENCES customer(id)
    );
3. **UNIQUE**
    1. Unique, can be null, table can have multiple unique attributes.
    2. CREATE TABLE customer (
    ...
    email VARCHAR(1024) UNIQUE,
    ...
    );
4. **CHECK**
    1. CREATE TABLE customer (
    ...
    CONSTRAINT age_check CHECK (age > 12),
    ...
    );
    2. "age_check", can also avoid this, MySQL generates name of constraint automatically.

5. **DEFAULT**
    1. Set default value of the column.
    2. CREATE TABLE account (
       ...
       saving-rate DOUBLE NOT NULL DEFAULT 4.25,
       ...
       );
6. An attribute can be **PK and FK both** in a table.
7. **ALTER OPERATIONS**
    1. Changes schema
    2. **ADD**
        1. **Add new column.**
        2. ALTER TABLE table_name ADD new_col_name datatype ADD new_col_name_2 datatype;
        3. e.g., ALTER TABLE customer ADD age INT NOT NULL;
    3. **MODIFY**
        1. **Change datatype of an attribute.**
        2. ALTER TABLE table-name MODIFY col-name col-datatype;
        3. E.g., VARCHAR TO CHAR
           ALTER TABLE customer MODIFY name CHAR(1024);
    4. **CHANGE COLUMN**
        1. **Rename column name.**
        2. ALTER TABLE table-name CHANGE COLUMN old-col-name new-col-name new-col-datatype;
        3. e.g., ALTER TABLE customer CHANGE COLUMN name customer-name VARCHAR(1024);
    5. **DROP COLUMN**
        1. **Drop a column completely.**
        2. ALTER TABLE table-name DROP COLUMN col-name;
        3. e.g., ALTER TABLE customer DROP COLUMN middle-name;
    6. **RENAME**
        1. **Rename table name itself.**
        2. ALTER TABLE table-name RENAME TO new-table-name;
        3. e.g., ALTER TABLE customer RENAME TO customer-details;

**DATA MANIPULATION LANGUAGE (DML)**

1. **INSERT**
    1. INSERT INTO table-name(col1, col2, col3) VALUES (v1, v2, v3), (val1, val2, val3);
2. **UPDATE**
    1. UPDATE table-name SET col1 = 1, col2 = 'abc' WHERE id = 1;
    2. Update multiple rows e.g.,
        1. UPDATE student SET standard = standard + 1;
    3. **ON UPDATE CASCADE**
        1. Can be added to the table while creating constraints. Suppose there is a situation where we have two tables such that primary key of one table is the foreign key for another table. if we update the primary key of the first table then using the ON UPDATE CASCADE foreign key of the second table automatically get updated.
3. **DELETE**
    1. DELETE FROM table-name WHERE id = 1;
    2. DELETE FROM table-name; //all rows will be deleted.
    3. **DELETE CASCADE** - (**to overcome DELETE constraint of Referential constraints**)
        1. What would happen to child entry if parent table's entry is deleted?
        2. CREATE TABLE ORDER (
           order_id int PRIMARY KEY,
           delivery_date DATE,
           cust_id INT,

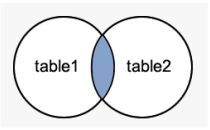
FOREIGN KEY(cust_id) REFERENCES customer(id) ON DELETE CASCADE
);

3. **ON DELETE NULL** - (**can FK have null values?**)
    1. CREATE TABLE ORDER (
    order_id int PRIMARY KEY,
    delivery_date DATE,
    cust_id INT,
    FOREIGN KEY(cust_id) REFERENCES customer(id) ON DELETE SET NULL
    );

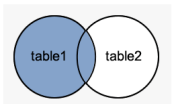
4. **REPLACE**
    1. Primarily used for already present tuple in a table.
    2. As UPDATE, using REPLACE with the help of WHERE clause in PK, then that row will be replaced.
    3. As INSERT, if there is no duplicate data new tuple will be inserted.
    4. REPLACE INTO student (id, class) VALUES(4, 3);
    5. REPLACE INTO table SET col1 = val1, col2 = val2;

**JOINING TABLES**

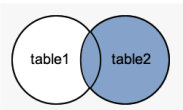
1. All **RDBMS** are relational in nature, we refer to other tables to get meaningful outcomes.
2. FK are used to do reference to other table.
3. **INNER JOIN**
    1. Returns a resultant table that has matching values from both the tables or all the tables.
    2. SELECT column-list FROM table1 INNER JOIN table2 ON condition1
    INNER JOIN table3 ON condition2
    ...;



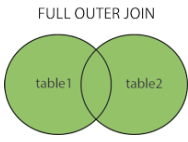

    3. **Alias in MySQL (AS)**
        1. Aliases in MySQL is used to give a temporary name to a table or a column in a table for the purpose of a particular query. It works as a nickname for expressing the tables or column names. It makes the query short and neat.
        2. SELECT col_name AS alias_name FROM table_name;
        3. SELECT col_name1, col_name2,... FROM table_name AS alias_name;
4. **OUTER JOIN**
    1. **LEFT JOIN**
        1. This returns a resulting table that all the data from left table and the matched data from the right table.
        2. SELECT columns FROM table LEFT JOIN table2 ON Join_Condition;



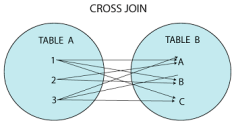

    2. **RIGHT JOIN**
        1. This returns a resulting table that all the data from right table and the matched data from the left table.
        2. SELECT columns FROM table RIGHT JOIN table2 ON join_cond;




    3. **FULL JOIN**
        1. This returns a resulting table that contains all data when there is a match on left or right table data.
        2. **Emulated** in MySQL using LEFT and RIGHT JOIN.
        3. LEFT JOIN UNION RIGHT JOIN.
        4. SELECT columns FROM table1 as t1 LEFT JOIN table2 as t2 ON t1.id = t2.id
        UNION
        SELECT columns FROM table1 as t1 RIGHT JOIN table2 as t2 ON t1.id = t2.id;
        5. UNION ALL, can also be used this will duplicate values as well while UNION gives unique values.




5. **CROSS JOIN**
    1. This returns all the cartesian products of the data present in both tables. Hence, all possible variations are reflected in the output.
    2. Used rarely in practical purpose.
    3. Table-1 has 10 rows and table-2 has 5, then resultant would have 50 rows.
    4. SELECT column-lists FROM table1 CROSS JOIN table2;




6. **SELF JOIN**

1. It is used to get the output from a particular table when the same table is joined to itself.
2. Used very less.
3. Emulated using INNER JOIN.
4. SELECT columns FROM table as t1 INNER JOIN table as t2 ON t1.id = t2.id;

**7. Join without using join keywords.**

1. SELECT * FROM table1, table2 WHERE condition;
2. e.g., SELECT artist_name, album_name, year_recordedFROM artist, albumWHERE artist.id = album.artist_id;

**SET OPERATIONS**

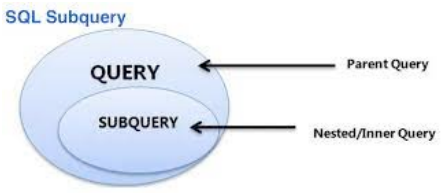
1. Used to combine multiple select statements.
2. Always gives distinct rows.

| JOIN | SET Operations |
|---|---|
| Combines multiple tables based on matching condition. | Combination is resulting set from two or more SELECT statements. |
| Column wise combination. | Row wise combination. |
| Data types of two tables can be different. | Datatypes of corresponding columns from each table should be the same. |
| Can generate both distinct or duplicate rows. | Generate distinct rows. |
| The number of column(s) selected may or may not be the same from each table. | The number of column(s) selected must be the same from each table. |
| Combines results horizontally. | Combines results vertically. |

3. **UNION**
   1. Combines two or more SELECT statements.
   2. SELECT * FROM table1
      UNION
      SELECT * FROM table2;
   3. Number of column, order of column must be same for table1 and table2.
4. **INTERSECT**
   1. Returns common values of the tables.
   2. Emulated.
   3. SELECT DISTINCT column-list FROM table-1 INNER JOIN table-2 USING(join_cond);
   4. SELECT DISTINCT * FROM table1 INNER JOIN table2 ON USING(id);
5. **MINUS**
   1. This operator returns the distinct row from the first table that does not occur in the second table.
   2. Emulated.
   3. SELECT column_list FROM table1 LEFT JOIN table2 ON condition WHERE table2.column_name IS NULL;
   4. e.g., SELECT id FROM table-1 LEFT JOIN table-2 USING(id) WHERE table-2.id IS NULL;

**SUB QUERIES**

1. Outer query depends on inner query.
2. Alternative to joins.
3. Nested queries.
4. SELECT column_list (s) FROM table_name WHERE column_name OPERATOR (SELECT column_list (s) FROM table_name [WHERE]);
5. e.g., SELECT * FROM table1 WHERE col1 IN (SELECT col1 FROM table1);
6. Sub queries exist mainly in 3 clauses
   1. Inside a WHERE clause.



2. Inside a FROM clause.
3. Inside a SELECT clause.

7. **Subquery using FROM clause**
   1. SELECT MAX(rating) FROM (SELECT * FROM movie WHERE country = 'India') as temp;
8. **Subquery using SELECT**
   1. SELECT (SELECT column_list(s) FROM T_name WHERE condition), columnList(s) FROM T2_name WHERE condition;
9. **Derived Subquery**
   1. SELECT columnLists(s) FROM (SELECT columnLists(s) FROM table_name WHERE [condition]) as new_table_name;
10. **Co-related sub-queries**
    1. With a normal nested subquery, the inner SELECT query runs first and executes once, returning values to be used by the main query. A correlated subquery, however, executes once for each candidate row considered by the outer query. In other words, the inner query is driven by the outer query.

```
SELECT column1, column2, ....
FROM table1 as outer
WHERE column1 operator
                    (SELECT column1, column2
                     FROM table2
                     WHERE expr1 =
                                 outer.expr2);
```

**JOIN VS SUB-QUERIES**

| JOINS | SUBQUERIES |
|---|---|
| Faster | Slower |
| Joins maximise calculation burden on DBMS | Keeps responsibility of calculation on user. |
| Complex, difficult to understand and implement | Comparatively easy to understand and implement. |
| Choosing optimal join for optimal use case is difficult | Easy. |

**MySQL VIEWS**

1. A view is a database object that has no values. Its contents are based on the base table. It contains rows and columns similar to the real table.
2. In MySQL, the View is a **virtual table** created by a query by joining one or more tables. It is operated similarly to the base table but does not contain any data of its own.
3. The View and table have one main difference that the views are definitions built on top of other tables (or views). If any changes occur in the underlying table, the same changes reflected in the View also.
4. CREATE VIEW view_name AS SELECT columns FROM tables [WHERE conditions];
5. ALTER VIEW view_name AS SELECT columns FROM table WHERE conditions;
6. DROP VIEW IF EXISTS view_name;
7. CREATE VIEW Trainer AS SELECT c.course_name, c.trainer, t.email FROM courses c, contact t WHERE c.id = t.id; (View using Join clause).

NOTE: We can also import/export table schema from files (.csv or json).