1.12.1(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 1.12.1 provides a comparison between M2M and IoT.

Table 1.12.1 : 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
Data collection and Analysis	Mostly Local	Mostly in the Cloud
Communication	Mostly point-to-point (OSI Layers 1-3)	Mostly through higher layer protocols (OSI Layers 4-7)

The Fig. 1.12.4 highlights the communication protocols, that you read about in Chapter 1, as used for M2M and IoT communication respectively.

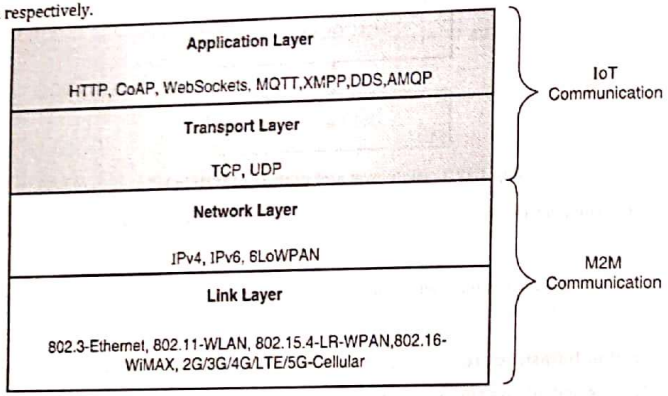

Fig. 1.12.4 : Communication protocols

1.13 The oneM2M IoT Standardised Architecture

There are several efforts that are undergoing for standardising IoT architectures and frameworks. oneM2M is one such M2M architecture standardisation approach. ONEM2M Technical Specification describes the functional architecture for the oneM2M Services Platform in the document number TS-0001-V4.5.0 released on 24-Feb-2020.

It takes the following layered model approach for supporting end-to-end (E2E) M2M Services.

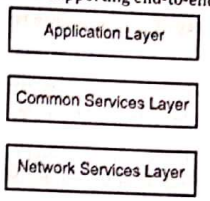

Fig. 1.13.1

The Fig. 1.13.2 illustrates the high-level oneM2M functional architecture.

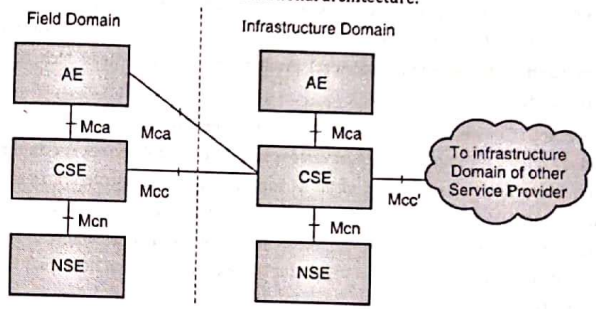

Fig. 1.13.2 : oneM2M functional architecture

The oneM2M functional architecture has the following functions :

1. **Application Entity (AE) :** Application Entity is an entity in the application layer that implements an M2M application service logic. Each application service logic can be resident in a number of M2M nodes and/or more than once on a single M2M node. Each execution instance of an application service logic is termed an "Application Entity" (AE) and is identified with a unique AE-ID. Examples of the AEs include an instance of a fleet tracking application, a remote blood sugar monitoring application, a power metering application, or a controlling application.
2. **Common Services Entity (CSE) :** A Common Services Entity represents an instantiation of a set of "common service functions" of the M2M environments. Such service functions are exposed to other entities through the Mca and Mcc reference points. Reference point Mcn is used for accessing underlying Network Service Entities. Each Common Service Entity is identified with a unique CSE-ID.

   Examples of service functions offered by CSE include: Data Management, Device Management, M2M Service Subscription Management, and Location Services. Such "sub-functions" offered by a CSE may be logically and informatively conceptualised as Common Services Functions (CSFs). The resources which implement the service functions in a CSE can be mandatory or optional.
3. **Underlying Network Services Entity (NSE) :** A Network Services Entity provides services from the underlying network to the CSEs. Examples of such services include device management, location services and device triggering. No particular organisation of the NSEs is assumed. Note here that the underlying networks provide data transport services between entities in the oneM2M System. Such data transport services are not included in the NSE.
4. **Reference Points :** A reference point consists of one or more interfaces of any kind. The following reference points are supported by the Common Services Entity (CSE).

The Mc(-) nomenclature is based on the mnemonic "M2M communications". So, wherever you read Mc_ in the reference point read Mc as "M2M communication". The reference points defined in the functional architecture are as following :

(a) Mca Reference Point : Communication flows between an Application Entity (AE) and a Common Services Entity (CSE) cross the Mca reference point. These flows enable the AE to use the services supported by the CSE, and for the CSE to communicate with the AE. Read Mca as "M2M communication application interface".

(b) Mcc Reference Point : Communication flows between two Common Services Entities (CSEs) cross the Mcc reference point. These flows enable a CSE to use the services supported by another CSE. Read Mcc as "M2M communication common service interface".

(c) Mcn Reference Point : Communication flows between a Common Services Entity (CSE) and the Network Services Entity (NSE) cross the Mcn reference point. These flows enable a CSE to use the supported services (other than transport and connectivity services) provided by the NSE. Read Mcn as "M2M communication network service interface".

(d) Mcc' Reference Point : Communication flows between two Common Services Entities (CSEs) in Infrastructure Nodes (IN) that are oneM2M compliant and that resides in different M2M Service Provider domains cross the Mcc' reference point. These flows enable a CSE of an IN residing in the Infrastructure Domain of an M2M Service Provider to communicate with a CSE of another IN residing in the Infrastructure Domain of another M2M Service Provider to use its supported services, and vice versa. Mcc' extends the reachability of services offered over the Mcc reference point, or a subset thereof. The trigger for these communication flows may be initiated elsewhere in the oneM2M network.

1.13.1 The IoT World Forum (IoTWF) Standardised Architecture

In 2014 the IoTWF architectural committee (led by Cisco, IBM, Rockwell Automation, and others) published a seven-layer IoT architectural reference model. It provides a concise way of visualising IoT from a technical perspective. Each of the seven layers is broken down into specific functions, and security encompasses the entire model.

The Fig. 1.13.3 outlines the seven layers in IoTWF standardised architecture.

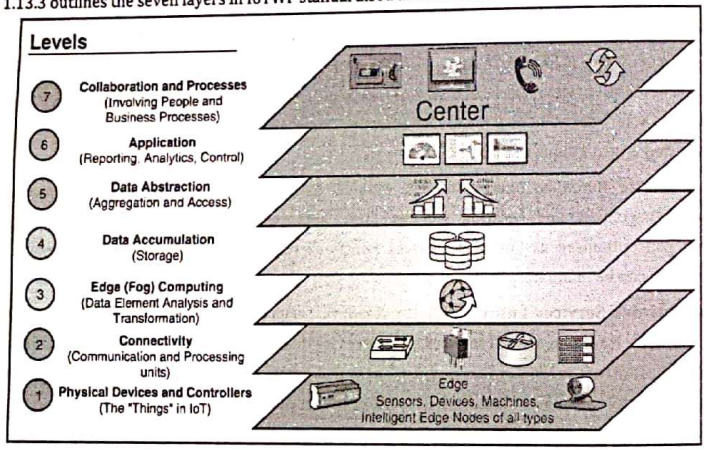

Fig. 1.13.3

This reference model

1. **Simplifies :** It helps to break down the complex systems so that each part is more understandable and individually approachable for defining interoperable solutions.
2. **Clarifies :** It provides additional information to precisely identify levels of the IoT and to establish common terminology at the respective layers.
3. **Identifies :** It identifies where specific types of processing is optimised across different parts of the system. You could choose to identify more optimal and flexible solutions at the respective layers.
4. **Standardises :** It provides a first step in enabling vendors to create IoT products that work with each other. Like the OSI model, the IoT reference model clearly articulates the nature of layers and what it could mean to build interoperable products.
5. **Organises :** It makes the IoT real and approachable, instead of simply conceptual. Various layers could be owned and managed by different teams within the same organisation.

Let's examine each of the seven layers briefly.

Level 1 : Physical Devices and Controllers

The first level of the IoT Reference Model is the physical devices and controllers level. At this level, you have "things" as in the Internet of Things. These are the "things" that include a wide range of endpoint devices that send and receive information. Their primary function is generating data. These "things" could be queried or controlled over a network remotely.

Today, the list of devices ("things") is already extensive. Devices are diverse, and there are no rules about size, location, form factor, or origin. Some devices could be the size of a silicon chip whereas some could be as large as vehicles. The IoT reference model supports the entire range of possible devices. Today, hundreds of equipment manufacturers produce IoT devices. To simplify compatibility and support manufacturability, the IoT Reference Model generally describes the level of processing needed from Level 1 devices. It is outlined in the Fig. 1.13.4.

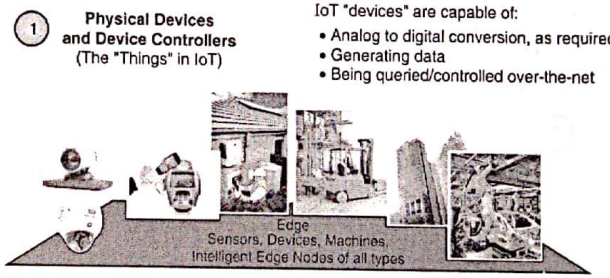

Fig. 1.13.4

Level 2 : Connectivity

In the second layer of the IoT Reference Model, the focus is on connectivity. The most important function of this IoT layer is the reliable and timely transmission of data. More specifically, this includes transmissions :

1. Between devices (Level 1) and the network
2. Across networks at Level 2

3. Between the network (Level 2) and low-level information processing occurring at Level 3

The IoT Reference Model does not require or indicate creation of a different network-it relies on existing networks. However, some legacy devices may not be IP-enabled, which could require introducing communication gateways.

The Fig. 1.13.5 outlines the functions of Level 2.

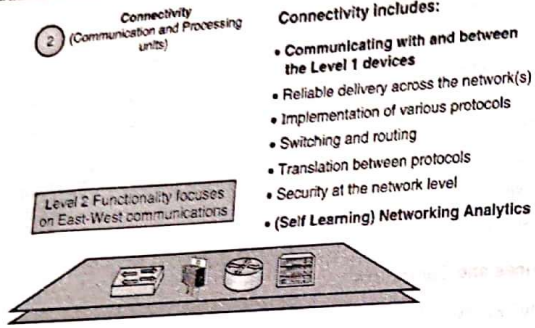

Fig. 1.13.5

Level 3 : Edge (Fog) Computing

Level 3 deals with edge computing. It is also referred as "fog" computing. At this layer, the emphasis is on data reduction and converting network data flows into information that is ready for storage and processing by higher layers. One of the basic principles of this reference model is that information processing is initiated as early and as close to the edge of the network as possible to avoid sending too much data to upper layers. The data processing should be done as close as to the source as possible. Another important function that occurs at Layer 3 is the evaluation of data to see if it can be filtered or aggregated before sending it to higher layers. This also allows for data to be reformatted or decoded, making additional processing by other systems easier. Thus, at this layer, another critical function is to assess the collected data from the devices and send alerts as appropriate.

The Fig. 1.13.6 outlines the functions of Level 3.

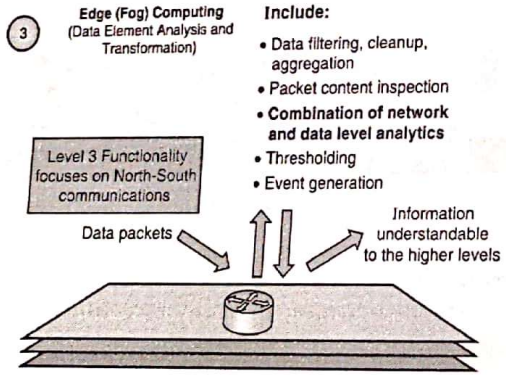

Fig. 1.13.6

Level 4 : Data Accumulation

Networking systems are built to reliably move data. The data is "in motion". Prior to Level 4, data is moving through the network at the rate and organisation determined by the devices generating the data. The data movement is event driven (on occurrence of something).

However, most applications do not need to process data in real-time as they occur. These applications typically use data "at rest" (stored data in memory or disk rather than working on continuously changing data). At Level 4, Data Accumulation, data in motion is converted to data at rest. Applications access the data when necessary. In short, Level 4 converts event-based data to query-based processing.

At a high-level, Level 4 determines

1. **Data usability requirements :** It determines whether data is of interest to higher levels. If yes, then Level 4 processing is the first level that is configured to serve the specific needs of a higher level.
2. **Data persistence requirements :** It determines whether data must be stored. If yes, then where it should be stored, in which format, at what granularity level, and for how long.
3. **Data accumulation requirements :** It determines whether data must be recombined, recomputed, or aggregated with previously stored information. Some of the data might have come from non-IoT sources.

The Fig. 1.13.7 outlines the functions of Level 4.

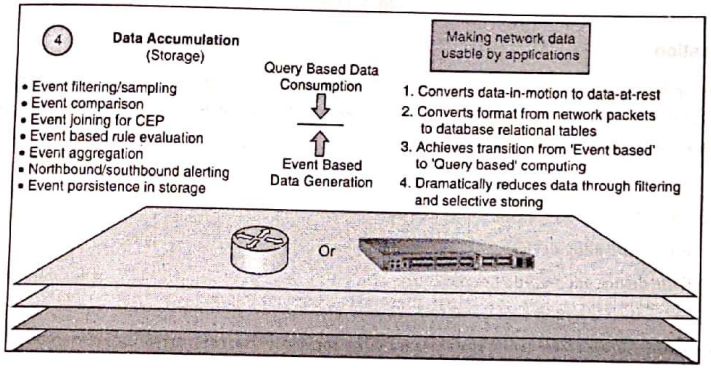

Fig. 1.13.7

Level 5 : Data Abstraction

IoT systems need to scale to a corporate or even global level and require multiple storage systems to accommodate IoT device data and data from traditional enterprise sources such as ERP, HRMS, CRM, and other systems. The data abstraction functions of Level 5 are focused on rendering data and its storage in ways that enable developing simpler and performance-enhanced applications.

At a high-level, the data abstraction level does the following.

1. Consolidate multiple data formats from different sources
2. Make semantics of data across sources consistent
3. Ensure data completeness from higher-level application perspective
4. Consolidate and visualise data into one place from different sources

5. Secure and protect data
6. Normalise, denormalise, and index data to provide fast application access

The Fig. 1.13.8 outlines the functions of Level 5.

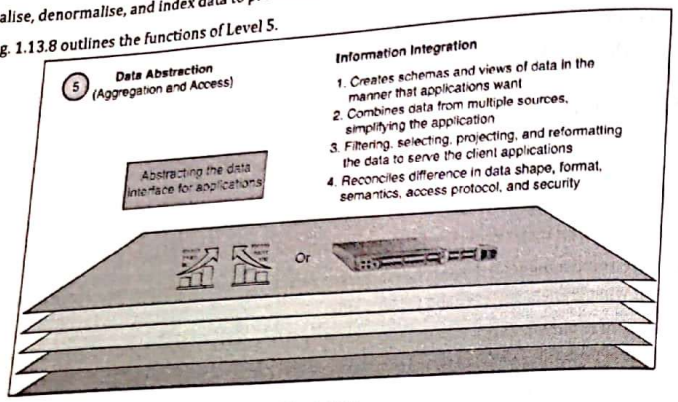

Fig. 1.13.8

Level 6 : Application

Level 6 is the application level, where information interpretation occurs. Software at this level interacts with Level 5 and data at rest, so it does not have to operate at network speeds. The IoT Reference Model does not strictly define an application. Applications vary based on requirements, the nature of device data, and business needs.

For example

- Some applications may focus on monitoring device data
- Some may focus on controlling devices
- Some may combine device and non-device data

Monitoring and control applications represent many different application models, programming patterns, and software, leading to discussions of operating systems, mobility, application servers, multi-threading, etc.

The Fig. 1.13.9 outlines the functions of Level 6.

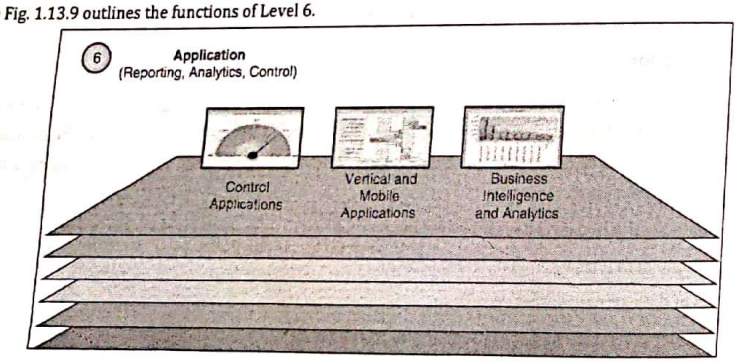

Fig. 1.13.9

Level 7 : Collaboration and Processes

The IoT system, and the information it creates, is of little value unless it yields action, which often requires people and processes. Applications execute business logic to empower people. People use applications and associated data for their specific needs. Often, multiple people use the same application for a range of different purposes. So the objective is not to just create the application but to empower people to do their work better. Applications (Level 6) give business people the right data, at the right time, so that they can do the right thing.

But frequently, the action needed requires more than one person. People must be able to communicate and collaborate, sometimes using the traditional Internet, to make the IoT useful. Communication and collaboration often requires multiple steps and it usually goes beyond multiple applications. Level 7 deals with collaboration and processes.

1.13.2 Security in the IoT

The IoT reference model also highlights security requirements at each of the seven levels of the IoT reference architecture. The security measures must

- Secure each device or system
- Provide security for all processes at each level
- Secure movement and communication between each level

The Fig. 1.13.10 outlines the security functions at all seven layers.

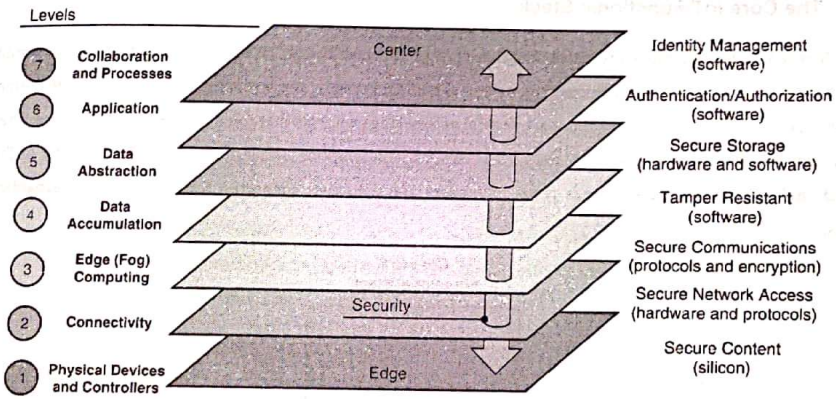

Fig. 1.13.10

1.14 A Simplified IoT Architecture (IoT Protocol Stack)

So far you have read about two IoT architectures. Similarly, there are various other standard development organisations that have attempted to define an architecture for IoT. While all of these architectures have their own pros and cons, for our in-general discussion in this chapter, let's look at a simplified IoT architecture that we can use for focusing on key aspects of the IoT architecture.

The Fig. 1.14.1 outlines a simplified IoT architecture.

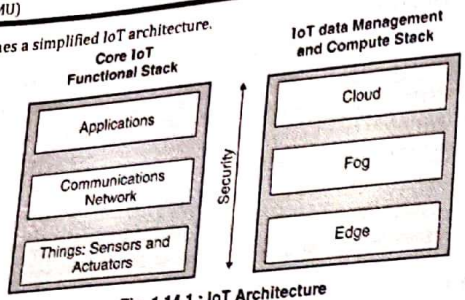

Fig. 1.14.1 : IoT Architecture

The simplified IoT architecture highlights the fundamental building blocks that are common to most IoT systems. The simplified IoT architecture could be thought of having two parallel stacks (layers) as

1. The Core IoT Functional Stack
2. The IoT Data Management and Compute Stack

Let's learn about these IoT stacks in further details.

1.14.1 The Core IoT Functional Stack

The Core IoT Functional Stack represents the three layers that simplify the IoT architecture into its most foundational building blocks. Each block in the stack could be further expanded based on the technologies you choose to implement the stack. For example, as you understand, there are several networking protocols that could be used in IoT systems. The entire OSI layer itself has seven layers that cover the various aspects of networking. Similarly, the bottom most layer of The Core IoT Functional Stack could have various types of sensors and actuators connected through various mechanisms and protocols.

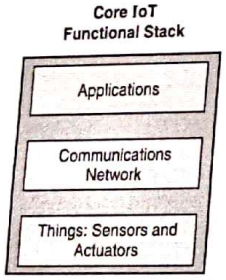

Fig. 1.14.2 : Core IoT Functional Stack

The three layers are further explained as following.

1. **"Things" Layer :** At this layer, the physical devices (sensors and actuators) are used as per the constraints of the environment in which they are deployed.

The physical devices carry out tasks such as information collection (through sensors) and taking actions (through actuators). Today, there are thousands of physical devices that could be used as per your requirements. They come in variety of types, shapes, functionalities, and operate using various protocols and mechanisms. You will learn more about them in the subsequent chapters.

2. **Communications Network Layer :** Communication layer is the heart of any IoT system. This is the layer that actually connects the physical devices to the external network and make them really "smart" and accessible over different networks. The communication network layer could use either wired or wireless medium to connect the physical devices to the network. However, wireless technologies are more common for IoT systems. This layer could have four sublayers as following.

   (a) Access network sublayer : Access network sublayer connects physical devices to the network. The physical devices could be directly accessed and operated using the access network sublayer. It is usually based off the wireless technologies such as Wi-Fi, ZigBee and Bluetooth. However, note here that the physical devices could also have this layer connected through the wired medium. For example, you could turn on a smart bulb in your home using home Wi-Fi network.

   (b) Gateways and backhaul network sublayer : The various access network sublayer endpoints could be connected to a common communication system called a gateway.

   If you have ever done a networking connection, you must be aware that each end point device on the network has a default gateway defined through which it can communicate to an external network.

   The devices on the same network (subnet) could talk to each other without the gateway. But, if the devices need to connect to other networks than its own network, then a gateway is required. For example, you can connect two phones, in close proximity, over Bluetooth and send data. But, if you need to send the data to a different phone, which is not in the local proximity, you will require to connect your phone to an external network, say a Wi-Fi, via a gateway. A common communication system organises multiple smart objects in a given area around a common gateway. The gateway communicates directly with the smart objects.

   The role of the gateway is to forward the collected information through a longer-range medium (called the backhaul) to a central information processing station where the information is further processed through several applications. For example, if you want to remotely control your smart bulb at home, you would require a connection mechanism to control the bulb over cellular or Wi-Fi network via an application that could connect you to the gateway setup at your home using which you can operate the bulb.

   (c) Network transport sublayer : This sublayer refers to the OSI model layers 3 and 4 where IP, TCP and UDP protocols are used for sending and receiving the information. The "things" could send the information via the gateway and could receive the control and management instructions via the gateway. For example, a temperature sensor could periodically send (via the gateway) temperature data to the cloud based storage system or application for processing.

(d) IoT network management sublayer : At this layer, additional networking protocols could be used to exchange data with "things" using lower level protocols such as TCP and UDP. Some of the examples of such higher level protocols could be CoAP, DDS, AMQP, and MQTT.

3. **Application and Analytics Layer :** At this layer, there could be several applications that process, analyse, visualise, control, manage and report the collected data from "things".

You could build several useful applications based on your requirements. Some of the applications could just control, manage and operate the IoT systems while others could synthesise the collected information and generate meaningful reports and actionable insights.

1.14.2 IoT Data Management and Compute Stack

Typically, how many lights do you think a shopping mall would have? How large the area should be to predict weather forecast sufficiently and confidently, a few hundred square kilometres? In a typical industrial plants, how many valves and switches would need to be operated? In a store like Amazon Go, how many items would be on the shelf and how many customers would visit the store every day? You may not know the exact number, but you would surely agree that we are talking about a very large scale - anywhere between thousands to millions of devices to be managed, controlled, and operated.

To give you a better perspective of scale, assume that you have 4 temperature sensors in your home that record temperature every 5 second (12 data points per minutes per device).

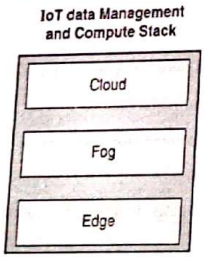

Fig. 1.14.3

So, in a day you would have $4 \times 12 \times 60 \times 24 = 69{,}120$ data points for just for 4 sensors! Imagine the volume of data that could be generated by a shopping mall where thousands of temperature sensors are fitted that monitor the entire shopping mall for any incidence such as fire. Just too much, isn't it?

To support massive scale requirements of IoT systems, new data management and computing stacks (architectures) are evolving. The new architecture not only needs to support an extremely large number of IoT devices from connection (and control) perspective but also should provide mechanisms for processing the huge volume of data that these IoT devices would likely produce every day.

The IoT devices themselves are low on processing power and other computing resources. Hence, the data produced by these devices are sent to external (more powerful) systems for storage and processing as required.

1.14.2(A) Why Sending All Data to The Cloud May Not Be The Obvious Choice? (Design Considerations and Data Related Problems)

You might argue that you could send the entire IoT system generated data to a central location, preferably cloud, and it could solve all the problem. But, that isn't always so simple, and neither is the preferable choice. Here is why.

1. **Latency :** Many IoT system, such as fire alarms and industrial plants, have requirements to respond almost in real-time (in millisecond). Any delay caused in sending and receiving data might prove to be disastrous. Sending all data to the cloud and then receiving response in milliseconds may not be possible.
2. **Network bandwidth :** As you understood earlier, IoT systems can generate huge amount of data. For example, commercial aeroplanes could generate 10 TB of data for every 30 minutes of flight. Transferring such a huge amount of data consumes too much network bandwidth. Also, it is not necessary to send the entire data to cloud because many critical analyses do not require cloud-scale processing and storage.

1.14.3 Fog Computing

So, as you understand, local processing and analysis over data could provide near real-time response and also conserve the valuable network bandwidth. But, how could you build such an architecture where only the data storage and processing that needs cloud resources are sent to it and remaining data processing and analysis occurs locally? That is precisely where Fog Computing comes into the picture.

Note : Don't worry too much about why Fog computing is called Fog. It is just an easily understood conceptual term. Drawing on the nature's analogy, cloud is in the sky farther from the earth whereas fog is closer to the earth. Similarly, cloud computing resources are distant and fog computing resources are closer to the actual device and network. Sometimes, technology can really have some cool and jazzy names!

As per NIST,

Definition : *Fog computing is a layered model that facilitates the deployment of distributed, latency-aware applications and services.*

Managing the data generated by IoT systems is one of the biggest challenges faced when deploying IoT systems. Fog computing addresses the challenge by decentralising applications, management, and data analytics into the network itself using a distributed compute model.

Fog nodes (physical or virtual) reside between smart IoT devices and centralised (cloud) services. The fog nodes are context aware and support a common data management and communication system. They can be organised in clusters as required. Fog computing minimises the request-response time from and to supported applications. When needed, fog computing also provides local computing resources and network connectivity to centralised services for the IoT devices.

The Fig. 1.14.4 outlines the fog computing layer that resides between the IoT devices and cloud services.

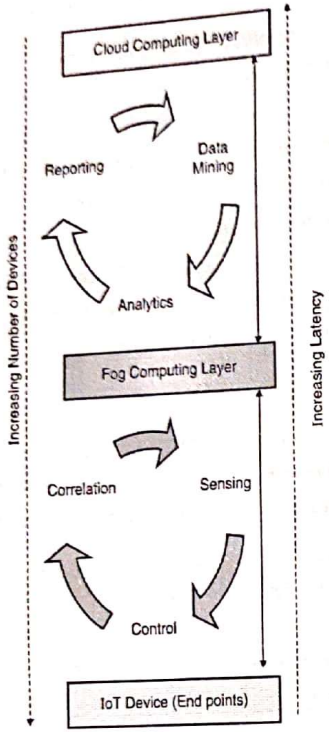

Fig. 1.14.4

As you see, the real-time functions such as sensing, control and correlation are carried out locally within the fog computing layer whereas resource intensive tasks such as data mining, analytics and reporting occur at the cloud computing layer.

Note here that fog computing is not perceived as a mandatory layer for such ecosystems nor is the centralised (cloud) service perceived as being required for a fog computing layer to support the functionality of smart end-devices. Different use case scenarios might have different architectures based on the optimal approach to supporting end-devices functionality.

1.14.3(A) Fog Node

The fog node is the core component of the fog computing architecture. Fog nodes could be either physical components, such as gateways, switches, routers, servers or virtual components, such as virtualised switches and virtual machines. Fog nodes are tightly coupled with the smart end-devices or access networks and provide computing resources to these devices.

For example, a fog node could be associated with the temperature sensor and could record temperature variations all day along. There could be minor variations in temperature that may not be useful to send to the cloud. As long as the variations are within the configured threshold limit, the data need not be sent to the cloud. When the temperature variation is such that it could possibly lead to an event, then the fog node could provide the real-time response as well as send the data to the cloud so that the other applications can be informed about the situation at hand.

A fog node is aware of its geographical distribution and logical location within the context of its cluster. Additionally, fog nodes provide some form of data management and communication services between network's edge layer where end-devices reside, and the fog computing service or the centralised (cloud) computing resources, when needed. To deploy a given fog computing capability, fog nodes operate in centralised or decentralised manner and can be configured as stand-alone fog nodes that communicate among them to deliver the service or can be combined to form clusters that provide horizontal scalability over disperse geolocations, through mirroring or extension mechanisms.

Characteristics of Fog Nodes

Fog nodes have the following characteristics.

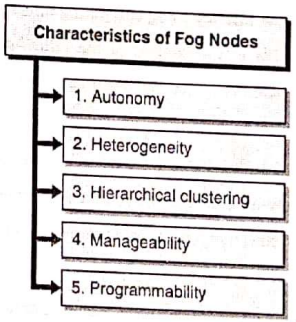

Fig. 1.14.5 : Characteristics of Fog Nodes

1. **Autonomy :** Fog nodes can operate independently, making local decisions, at the node or cluster-of-nodes level.
2. **Heterogeneity :** Fog nodes come in different form factors and can be deployed in a wide variety of environments.
3. **Hierarchical clustering :** Fog nodes support hierarchical structures, with different layers providing different subsets of service functions while working together as a continuum.
4. **Manageability :** Fog nodes are managed and orchestrated by complex systems that can perform most routine operations automatically.
5. **Programmability :** Fog nodes are inherently programmable at multiple levels, by multiple stakeholders - such as network operators, domain experts, equipment providers, or end users.

Characteristics of Fog Computing

Fog computing has the following characteristics.

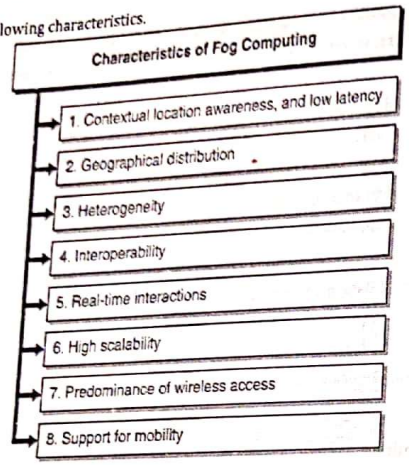

Fig. 1.14.6 : Characteristics of Fog Computing

1. **Contextual location awareness and low latency :** Fog computing offers the lowest-possible latency due to the fog nodes' awareness of their logical location in the context of the entire system and of the latency information for communicating with other nodes. Because fog nodes are often co-located with the smart end-devices, analysis and response to data generated by these devices is much quicker than from a centralised cloud service or data center.
2. **Geographical distribution :** Unlike centralised cloud services, the services and applications targeted by the fog computing are widely but geographically-identifiable distributed deployments. For example, fog computing plays an active role in delivering high quality streaming services to moving vehicles, through proxies and access points geographically positioned along highways and tracks.
3. **Heterogeneity :** Fog computing supports collection and processing of data of different form factors acquired through multiple types of network communication capabilities.
4. **Interoperability :** Fog computing components can interoperate with devices from multiple vendors and provide services for across various domains.
5. **Real-time interactions :** Fog computing applications involve real-time interactions rather than batch processing. Their proximity to end-devices make them suitable for providing real-time responses.
6. **High scalability :** Fog computing layer could be massively scalable. It can adapt its scale at a cluster or cluster-of-clusters level, supporting flexible computing, resource pooling, data-load changes, and various network conditions.

7. **Predominance of wireless access :** Although fog computing is used in wired environments, the large scale of wireless sensors in IoT demand distributed analytics and compute. For this reason, fog computing is very well suited to wireless IoT access networks.
8. **Support for mobility :** Many fog computing applications communicate directly with mobile devices, and therefore support mobility techniques, such as the location based services.

1.14.4 Edge Computing

IoT devices and sensors typically have constrained resources. However, as there are further enhancements in silicon technology, the computing capabilities on even tiny devices are becoming quite powerful and could be handy to perform computation even further closer to actual data generating device than fog computing.

The end-devices could perform some low-level computing tasks such as taking smart decisions by performing local analysis. This shift of computing to the actual end-device is called edge computing (as computing is occurring right on the end-device itself).

Definition : *Edge computing is the network layer encompassing the end-devices and their users.*

Edge computing provides local computing capability on a sensor, metering or some other end-devices that are network-accessible. This peripheral layer is also often referred to as IoT network.

1.14.4(A) The Hierarchy of Edge, Fog and Cloud

Alright, so now you understand that Edge, Fog and Cloud make the hierarchy that you started with in this section.

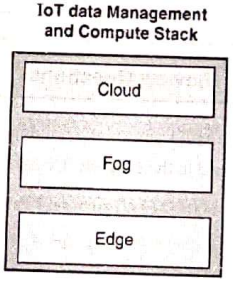

Fig. 1.14.7

Edge is the bottom most layer, Fog is the middle layer, and the Cloud is the top most layer. Note here that each of these layers can independently exist of its own. Edge and Fog layers do not replace the Cloud layer. Each layer has its own capabilities, and you may choose their functions as per your requirements. Functions requiring time-sensitive data processing and response should be as close to the device as possible – mostly at the edge or fog layer. Data processing requirements that involve analytics, visualisation, reporting and other higher value functions, that require significant computing resources, should be preferably addressed in the cloud.

1.14.5 Comparison between Edge, Fog, and Cloud Computing

Fog computing is often erroneously confused with edge computing, but there are key differences between the two concepts. The Table 1.14.1 summarises the key comparisons between Edge, Fog, and Cloud computing.

Table 1.14.1

Comparison Attributes	Edge Computing	Fog Computing	Cloud Computing
App Hosting	No	Yes	Yes
Flexible computing power	No	Yes	Yes
Resource Pooling	No	Yes	Yes
Real-time response	Yes	Yes	No
Fault Tolerance	No	Yes	Yes
Device dependent	Yes	No	No
Entire domain awareness	No	Yes	Yes
Cloud awareness	No	Yes	Yes
Controllers	Specific to edge	General	General
Security Scope	Limited to device	Up to Fog Layer	Up to Cloud Layer
Big Data Analytics	No	No	Yes
Scalability	Low	Medium	High
Use of virtualisation	No	Yes	Yes
Data Storage	No	Yes, but limited	Yes, Nearly unlimited

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] 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 Write a short note on IoT and digitisation specifically highlighting the key differences between the terms. **[4 Marks]**

Q. 10 Write a short note on economic significance of IoT. **[4 Marks]**

Q. 11 Describe a few ways in which connected roadways could make a difference in future. **[4 Marks]**

Q. 12 Describe a few ways in which connected factories could make a difference in future. **[4 Marks]**

Q. 13 Describe a few ways in which smart connected buildings could make a difference in future. **[4 Marks]**

Q. 14 Describe a few ways in which smart creatures could make a difference in future. **[4 Marks]**

Q. 15 Write a short note on convergence of IT (Information Technology) and OT (Operational Technology). **[4 Marks]**

Q. 16 Compare IT (Information Technology) and OT (Operational Technology). **[4 Marks]**

Q. 17 With a block diagram, explain technical building blocks of IoT. **[6 Marks]**

Q. 18 With a block diagram, explain high-level architecture of IoT. **[6 Marks]**

Q. 19 Write a short note on IoT gateway. **[4 Marks]**

Q. 20 Explain the major issues and challenges with IoT. **[6 Marks]**

[B] IoT and M2M

Q. 21 Write a short note on M2M. **[4 Marks]**

Q. 22 Describe a few applications of M2M. **[4 Marks]**

Q. 23 With a block diagram, explain high-level architecture of M2M. **[6 Marks]**

Q. 24 Explain high-level architecture of M2M for IoT. **[6 Marks]**

Q. 25 Compare IoT and M2M. **[6 Marks]**

Q. 26 List the protocols used in M2M and IoT. **[4 Marks]**

[C] The oneM2M IoT Standardised Architecture

Q. 27 With a block diagram, explain The oneM2M IoT Standardised Architecture. **[6 Marks]**

Q. 28 Write a short note on Common Services Entity (CSE) with respect to The oneM2M IoT Standardised Architecture. **[4 Marks]**

Q. 29 Explain the Reference Points used in The oneM2M IoT Standardised Architecture. **[4 Marks]**

[D] The IoT World Forum (IoTWF) Standardised Architecture

Q. 30 Draw a conceptual diagram for The IoT World Forum (IoTWF) Standardised Architecture. **[4 Marks]**

Q. 31 What are the benefits of using The IoT World Forum (IoTWF) Standardised Architecture. **[4 Marks]**

Q. 32 Explain Level 1: Physical Devices and Controllers with respect to the IoT World Forum (IoTWF) Standardised Architecture. **[4 Marks]**

Q. 33 Explain Level 2: Connectivity with respect to the IoT World Forum (IoTWF) Standardised Architecture. **[4 Marks]**

Q. 34 Explain Level 3: Edge (Fog) Computing with respect to the IoT World Forum (IoTWF) Standardised Architecture. **[4 Marks]**

Q. 35 Explain Level 4: Data Accumulation with respect to the IoT World Forum (IoTWF) Standardised Architecture. **[4 Marks]**

Q. 36 Explain Level 5: Data Abstraction with respect to the IoT World Forum (IoTWF) Standardised Architecture. **[4 Marks]**

Q. 37 Explain Level 6: Application with respect to the IoT World Forum (IoTWF) Standardised Architecture. **[4 Marks]**

Q. 38 Explain Level 7: Collaboration and Processes with respect to the IoT World Forum (IoTWF) Standardised Architecture. **[4 Marks]**