MODULE 3: IOT PROCESSING TOPOLOGIES AND TYPES

3.1. DATA FORMAT

The Internet is a vast space where huge quantities and varieties of data are generated regularly and flow freely. As of January 2018, there are a reported 4:021 billion Internet users worldwide. The massive volume of data generated by this huge number of users is further enhanced by the multiple devices utilized by most users. In addition to these data-generating sources, non-human data generation sources such as sensor nodes and automated monitoring systems further add to the data load on the Internet. This huge data volume is composed of a variety of data such as e-mails, text documents (Word docs, PDFs, and others), social media posts, videos, audio files, and images, as shown in Figure 3.1. However, these data can be broadly grouped into two types based on how they can be accessed and stored: 1) Structured data and 2) unstructured data.

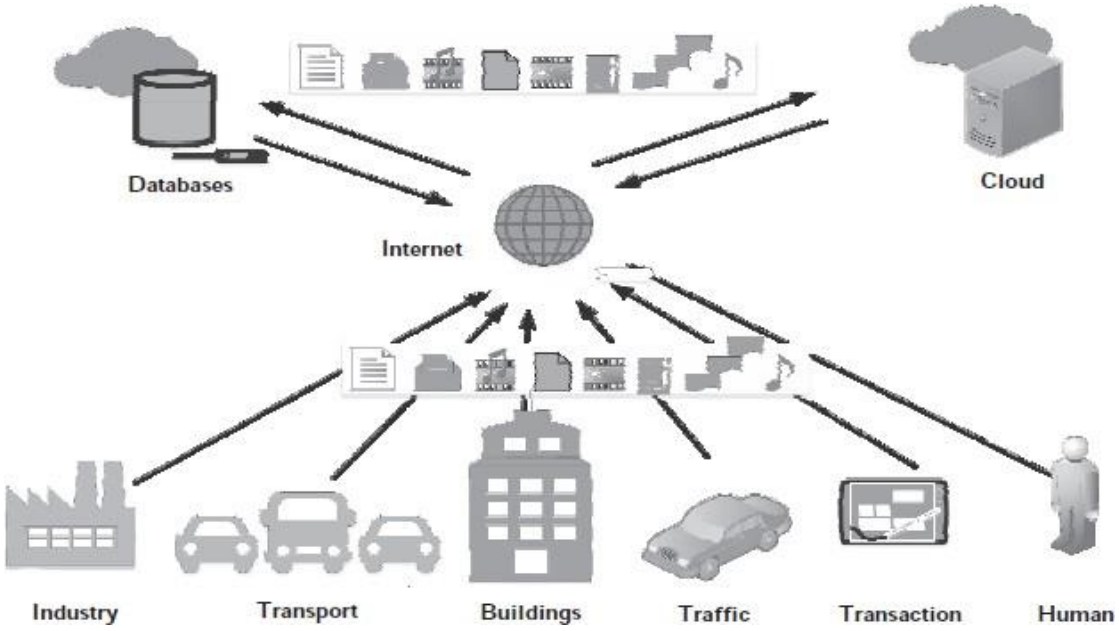

Figure 3.1 The various data generating and storage sources connected to the Internet and the plethora of data types contained within it

- **Structured data**

These are typically text data that have a pre-defined structure. Structured data are associated with relational database management systems (RDBMS). These are primarily created by using length-limited data fields such as phone numbers, social security numbers, and other such information. Even if the data is human or machine generated, these data are easily searchable by querying algorithms as well as human generated queries. Common usage of this type of data is associated with flight or train reservation systems, banking systems, inventory controls, and other similar systems. Established languages such as Structured Query Language (SQL) are used for accessing these data in RDBMS. However, in the context of IoT, structured data holds a minor share of the total generated data over the Internet.

- **Unstructured data**

In simple words, all the data on the Internet, which is not structured, is categorized as unstructured. These data types have no pre-defined structure and can vary according to

applications and data-generating sources. Some of the common examples of human-generated unstructured data include text, e-mails, videos, images, phone recordings, chats, and others. Some common examples of machine-generated unstructured data include sensor data from traffic, buildings, industries, satellite imagery, surveillance videos, and others. As already evident from its examples, this data type does not have fixed formats associated with it, which makes it very difficult for querying algorithms to perform a look-up. Querying languages such as NoSQL are generally used for this data type.

3.2. IMPORTANCE OF PROCESSING IN IOT

The vast amount and types of data flowing through the Internet necessitate the need for intelligent and resourceful processing techniques. This necessity has become even more crucial with the rapid advancements in IoT, which is laying enormous pressure on the existing network infrastructure globally. Given these urgencies, it is important to decide when to process and what to process? Before deciding upon the processing to pursue, we first divide the data to be processed into three types based on the urgency of processing: 1) Very time critical, 2) time critical, and 3) normal. Data from sources such as flight control systems, healthcare, and other such sources, which need immediate decision support, are deemed as very critical. These data have a very low threshold of processing latency, typically in the range of a few milliseconds. Data from sources that can tolerate normal processing latency are deemed as time critical data. These data, generally associated with sources such as vehicles, traffic, machine systems, smart home systems, surveillance systems, and others, which can tolerate a latency of a few seconds fall in this category. Finally, the last category of data, normal data, can tolerate a processing latency of a few minutes to a few hours and are typically associated with less data-sensitive domains such as agriculture, environmental monitoring, and others. Considering the requirements of data processing, the processing requirements of data from very time-critical sources are exceptionally high. Here, the need for processing the data in place or almost nearer to the source is crucial in achieving the deployment success of such domains. Similarly, considering the requirements of processing from category 2 data sources (time-critical), the processing requirements allow for the transmission of data to be processed to remote locations/processors such as clouds or through collaborative processing. Finally, the last category of data sources (normal) typically have no particular time requirements for processing urgently and are pursued leisurely as such.

3.3. PROCESSING TOPOLOGIES

The identification and intelligent selection of processing requirement of an IoT application are one of the crucial steps in deciding the architecture of the deployment. A properly designed IoT architecture would result in massive savings in network bandwidth and conserve significant amounts of overall energy in the architecture while providing the proper and allowable processing latencies for the solutions associated with the architecture. Regarding the importance of processing in IoT as outlined in Section 3.2, we can divide the various processing solutions into two large topologies: 1) On-site and 2) Off-site. The off-site processing topology can be further divided into the following: 1) Remote processing and 2) Collaborative processing.

- **On-site processing**

As evident from the name, the on-site processing topology signifies that the data is processed at the source itself. This is crucial in applications that have a very low tolerance for latencies. These latencies may result from the processing hardware or the network (during transmission of the data for processing away from the processor). Applications such as those associated with healthcare and flight control systems (real-time systems) have a breakneck data generation rate.

These additionally show rapid temporal changes that can be missed (leading to catastrophic damages) unless the processing infrastructure is fast and robust enough to handle such data. Figure 3.2 shows the on-site processing topology, where an event (here, fire) is detected utilizing a temperature sensor connected to a sensor node. The sensor node processes the information from the sensed event and generates an alert. The node additionally has the option of forwarding the data to a remote infrastructure for further analysis and storage.

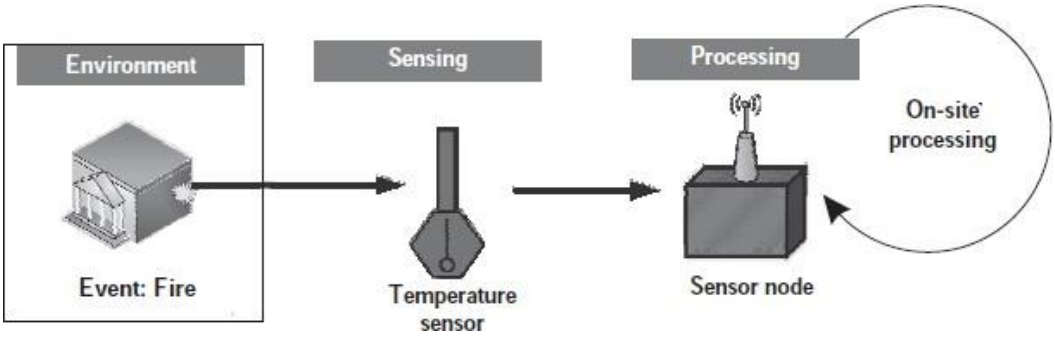

Figure 3.2 Event detection using an on-site processing topology

- **Off-site processing**

The off-site processing paradigm, as opposed to the on-site processing paradigms, allows for latencies (due to processing or network latencies); it is significantly cheaper than on-site processing topologies. This difference in cost is mainly due to the low demands and requirements of processing at the source itself. Often, the sensor nodes are not required to process data on an urgent basis, so having a dedicated and expensive on-site processing infrastructure is not sustainable for large-scale deployments typical of IoT deployments. In the off-site processing topology, the sensor node is responsible for the collection and framing of data that is eventually to be transmitted to another location for processing. Unlike the on-site processing topology, the off-site topology has a few dedicated high-processing enabled devices, which can be borrowed by multiple simpler sensor nodes to accomplish their tasks. At the same time, this arrangement keeps the costs of large-scale deployments extremely manageable. In the off-site topology, the data from these sensor nodes (data generating sources) is transmitted either to a remote location (which can either be a server or a cloud) or to multiple processing nodes. Multiple nodes can come together to share their processing power in order to collaboratively process the data (which is important in case a feasible communication pathway or connection to a remote location cannot be established by a single node).

- **Remote processing**

This is one of the most common processing topologies prevalent in present-day IoT solutions. It encompasses sensing of data by various sensor nodes; the data is then forwarded to a remote server or a cloud-based infrastructure for further processing and analytics. The processing of data from hundreds and thousands of sensor nodes can be simultaneously offloaded to a single, powerful computing platform; this results in massive cost and energy savings by enabling the reuse and reallocation of the same processing resource while also enabling the deployment of smaller and simpler processing nodes at the site of deployment. This setup also ensures massive scalability of solutions, without significantly affecting the cost of the deployment. Figure 3.3 shows the outline of one such paradigm, where the sensing of an event is performed locally, and the decision making is outsourced to a remote processor (here, cloud). However, this paradigm tends to use up a lot of network bandwidth and relies heavily on the presence of network connectivity between the sensor nodes and the remote processing infrastructure.

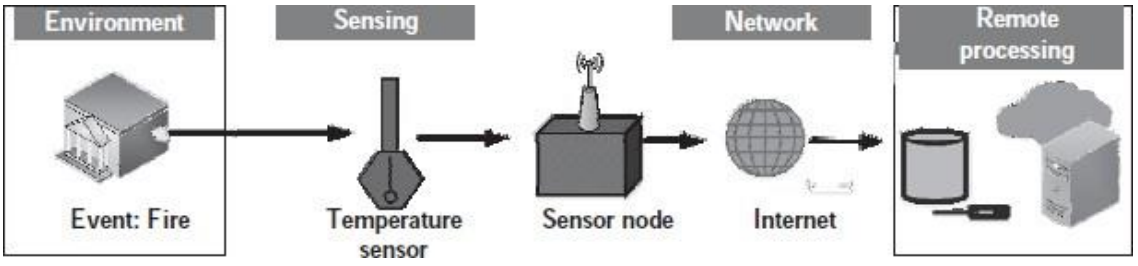

Figure 3.3 Event detection using an off-site remote processing topology

- **Collaborative processing**

This processing topology typically finds use in scenarios with limited or no network connectivity, especially systems lacking a backbone network. Additionally, this topology can be quite economical for large-scale deployments spread over vast areas, where providing networked access to a remote infrastructure is not viable. In such scenarios, the simplest solution is to club together the processing power of nearby processing nodes and collaboratively process the data in the vicinity of the data source itself. This approach also reduces latencies due to the transfer of data over the network. Additionally, it conserves bandwidth of the network, especially ones connecting to the Internet.

Figure 3.4 shows the collaborative processing topology for collaboratively processing data locally. This topology can be quite beneficial for applications such as agriculture, where an intense and temporally high frequency of data processing is not required as agricultural data is generally logged after significantly long intervals (in the range of hours). One important point to mention about this topology is the preference of mesh networks for easy implementation of this topology.

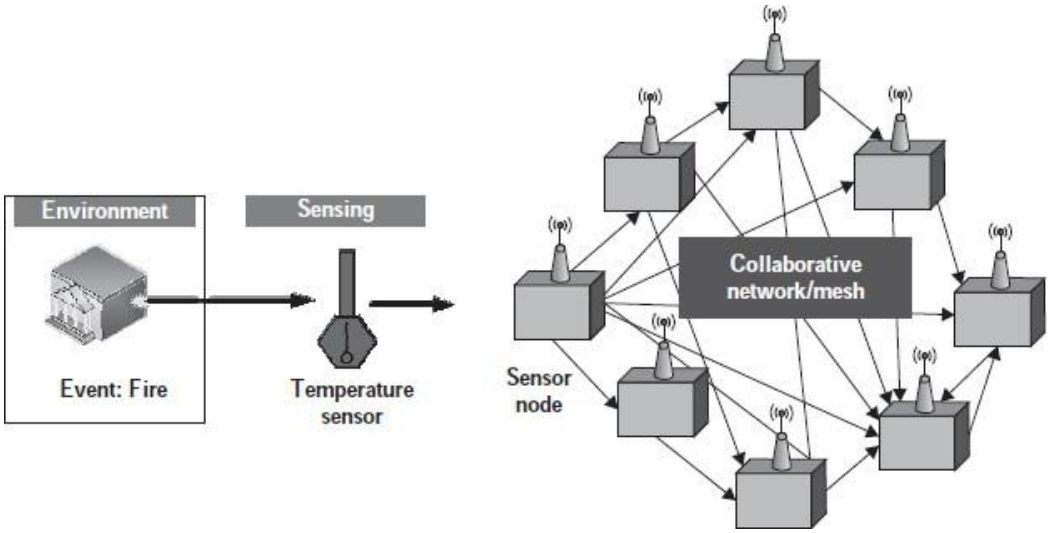

Figure 3.4 Event detection using a collaborative processing topology

3.4. IOT DEVICE DESIGN AND SELECTION CONSIDERATIONS

The main consideration of minutely defining an IoT solution is the selection of the processor for developing the sensing solution (i.e., the sensor node). This selection is governed by many parameters that affect the usability, design, and affordability of the designed IoT sensing and processing solution. In this chapter, we mainly focus on the deciding factors for selecting a processor for the design of a sensor node. The main factor governing the IoT device design and

selection for various applications is the processor. However, the other important considerations are as follows.

Size: This is one of the crucial factors for deciding the form factor and the energy consumption of a sensor node. It has been observed that larger the form factor, larger is the energy consumption of the hardware. Additionally, large form factors are not suitable for a significant bulk of IoT applications, which rely on minimal form factor solutions (e.g., wearables).

Energy: The energy requirements of a processor is the most important deciding factor in designing IoT-based sensing solutions. Higher the energy requirements, higher is the energy source (battery) replacement frequency. This principle automatically lowers the long-term sustainability of sensing hardware, especially for IoT-based applications.

Cost: The cost of a processor, besides the cost of sensors, is the driving force in deciding the density of deployment of sensor nodes for IoT-based solutions. Cheaper cost of the hardware enables a much higher density of hardware deployment by users of an IoT solution. For example, cheaper gas and fire detection solutions would enable users to include much more sensing hardware for a lesser cost.

Memory: The memory requirements (both volatile and non-volatile memory) of IoT devices determines the capabilities the device can be armed with. Features such as local data processing, data storage, data filtering, data formatting, and a host of other features rely heavily on the memory capabilities of devices. However, devices with higher memory tend to be costlier for obvious reasons.

Processing power: As covered in earlier sections, processing power is vital (comparable to memory) in deciding what type of sensors can be accommodated with the IoT device/node, and what processing features can integrate on-site with the IoT device. The processing power also decides the type of applications the device can be associated with. Typically, applications that handle video and image data require IoT devices with higher processing power as compared to applications requiring simple sensing of the environment.

I/O rating: The input–output (I/O) rating of IoT device, primarily the processor, is the deciding factor in determining the circuit complexity, energy usage, and requirements for support of various sensing solutions and sensor types. Newer processors have a meager I/O voltage rating of 3.3 V, as compared to 5 V for the somewhat older processors. This translates to requiring additional voltage and logic conversion circuitry to interface legacy technologies and sensors with the newer processors. Despite low power consumption due to reduced I/O voltage levels, this additional voltage and circuitry not only affects the complexity of the circuits but also affects the costs.

Add-ons: The support of various add-ons a processor or for that matter, an IoT device provides, such as analog to digital conversion (ADC) units, in-built clock circuits, connections to USB and ethernet, inbuilt wireless access capabilities, and others helps in defining the robustness and usability of a processor or IoT device in various application scenarios. Additionally, the provision for these add-ons also decides how fast a solution can be developed, especially the hardware part of the whole IoT application. As interfacing and integration of systems at the circuit level can be daunting to the uninitiated, the prior presence of these options with the processor makes the processor or device highly lucrative to the users/ developers.

3.5. PROCESSING OFFLOADING

The processing offloading paradigm is important for the development of densely deployable, energy-conserving, miniaturized, and cheap IoT-based solutions for sensing tasks. Building upon the basics of the off-site processing topology covered in the previous sections in this chapter, we delve a bit further into the various nuances of processing offloading in IoT.

Figure 3.5 shows the typical outline of an IoT deployment with the various layers of processing that are encountered spanning vastly different application domains from as near as sensing the

environment to as far as cloud-based infrastructure. Starting from the primary layer of sensing, we can have multiple sensing types tasked with detecting an environment (fire, surveillance, and others). The sensors enabling these sensing types are integrated with a processor using wired or wireless connections (mostly, wired). In the event that certain applications require immediate processing of the sensed data, an on-site processing topology is followed, similar to the one in Figure 3.2. However, for the majority of IoT applications, the bulk of the processing is carried out remotely in order to keep the on-site devices simple, small, and economical. Typically, for off-site processing, data from the sensing layer can be forwarded to the fog or cloud or can be contained within the edge layer. The edge layer makes use of devices within the local network to process data that which is similar to the collaborative processing topology shown in Figure 3.4. The devices within the local network, till the fog, generally communicate using short-range wireless connections. In case the data needs to be sent further up the chain to the cloud, long-range wireless connection enabling access to a backbone network is essential. Fog-based processing is still considered local because the fog nodes are typically localized within a geographic area and serve the IoT nodes within a much smaller coverage area as compared to the cloud. Fog nodes, which are at the level of gateways, may or may not be accessed by the IoT devices through the Internet.

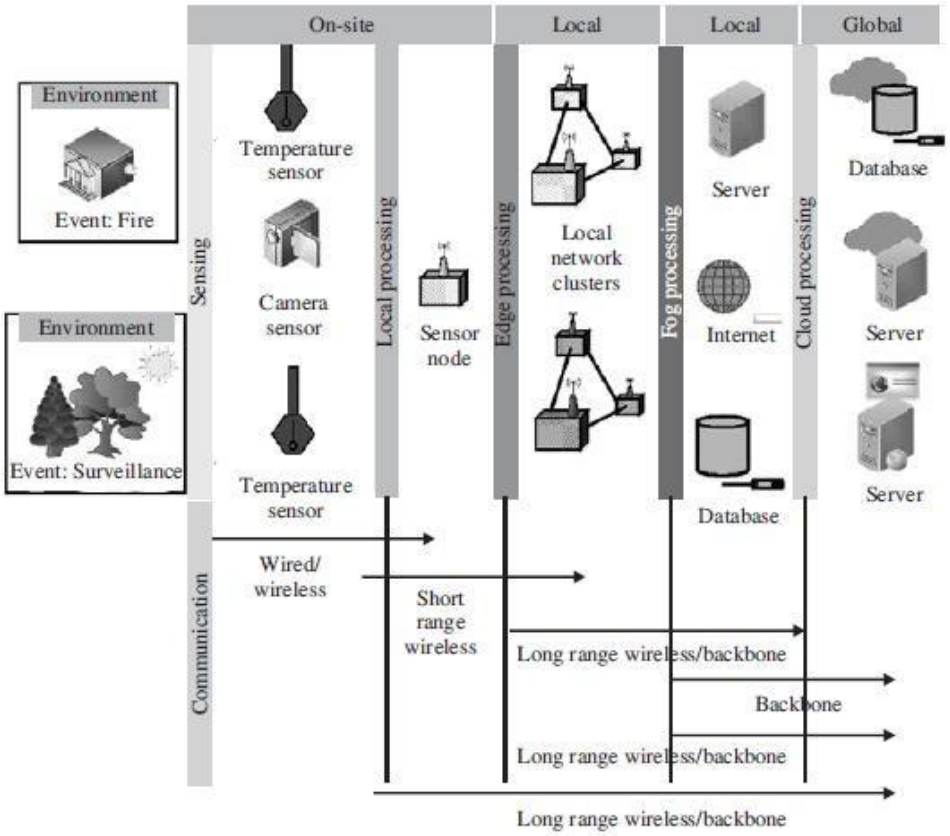

Figure 3.5 The various data generating and storage sources connected to the Internet and the plethora of data types contained within it

Finally, the approach of forwarding data to a cloud or a remote server, as shown in the topology in Figure 3.3, requires the devices to be connected to the Internet through long-range wireless/wired networks, which eventually connect to a backbone network. This approach is generally costly concerning network bandwidth, latency, as well as the complexity of the devices and the network infrastructure involved. This section on data offloading is divided into

three parts: 1) offload location (which outlines where all the processing can be offloaded in the IoT architecture), 2) offload decision making (how to choose where to offload the processing to and by how much), and finally 3) offloading considerations (deciding when to offload).

- **Offload location**

The choice of offload location decides the applicability, cost, and sustainability of the IoT application and deployment. We distinguish the offload location into four types:

Edge: Offloading processing to the edge implies that the data processing is facilitated to a location at or near the source of data generation itself. Offloading to the edge is done to achieve aggregation, manipulation, bandwidth reduction, and other data operations directly on an IoT device.

Fog: Fog computing is a decentralized computing infrastructure that is utilized to conserve network bandwidth, reduce latencies, restrict the amount of data unnecessarily flowing through the Internet, and enable rapid mobility support for IoT devices. The data, computing, storage and applications are shifted to a place between the data source and the cloud resulting in significantly reduced latencies and network bandwidth usage.

Remote Server: A simple remote server with good processing power may be used with IoT based applications to offload the processing from resource constrained IoT devices. Rapid scalability may be an issue with remote servers, and they may be costlier and hard to maintain in comparison to solutions such as the cloud.

Cloud: Cloud computing is a configurable computer system, which can get access to configurable resources, platforms, and high-level services through a shared pool hosted remotely. A cloud is provisioned for processing offloading so that processing resources can be rapidly provisioned with minimal effort over the Internet, which can be accessed globally. Cloud enables massive scalability of solutions as they can enable resource enhancement allocated to a user or solution in an on-demand manner, without the user having to go through the pains of acquiring and configuring new and costly hardware.

- **Offload decision making**

The choice of where to offload and how much to offload is one of the major deciding factors in the deployment of an offsite-processing topology-based IoT deployment architecture. The decision making is generally addressed considering data generation rate, network bandwidth, the criticality of applications, processing resource available at the offload site, and other factors. Some of these approaches are as follows.

Naive Approach: This approach is typically a hard approach, without too much decision making. It can be considered as a rule-based approach in which the data from IoT devices are offloaded to the nearest location based on the achievement of certain offload criteria. Although easy to implement, this approach is never recommended, especially for dense deployments, or deployments where the data generation rate is high or the data being offloaded in complex to handle (multimedia or hybrid data types). Generally, statistical measures are consulted for generating the rules for offload decision making.

Bargaining based approach: This approach, although a bit processing-intensive during the decision making stages, enables the alleviation of network traffic congestion, enhances service QoS (quality of service) parameters such as bandwidth, latencies, and others. At times, while trying to maximize multiple parameters for the whole IoT implementation, in order to provide the most optimal solution or QoS, not all parameters can be treated with equal importance. Bargaining based solutions try to maximize the QoS by trying to reach a point where the qualities of certain parameters are reduced, while the others are enhanced. This measure is

undertaken so that the achieved QoS is collaboratively better for the full implementation rather than a select few devices enjoying very high QoS. Game theory is a common example of the bargaining based approach. This approach does not need to depend on historical data for decision making purposes.

Learning based approach: Unlike the bargaining based approaches, the learning based approaches generally rely on past behavior and trends of data flow through the IoT architecture. The optimization of QoS parameters is pursued by learning from historical trends and trying to optimize previous solutions further and enhance the collective behavior of the IoT implementation. The memory requirements and processing requirements are high during the decision making stages. The most common example of a learning based approach is machine learning.

- **Offloading considerations**

There are a few offloading parameters which need to be considered while deciding upon the offloading type to choose. These considerations typically arise from the nature of the IoT application and the hardware being used to interact with the application. Some of these parameters are as follows.

Bandwidth: The maximum amount of data that can be simultaneously transmitted over the network between two points is the bandwidth of that network. The bandwidth of a wired or wireless network is also considered to be its data-carrying capacity and often used to describe the data rate of that network.

Latency: It is the time delay incurred between the start and completion of an operation. In the present context, latency can be due to the network (network latency) or the processor (processing latency). In either case, latency arises due to the physical limitations of the infrastructure, which is associated with an operation. The operation can be data transfer over a network or processing of a data at a processor.

Criticality: It defines the importance of a task being pursued by an IoT application. The more critical a task is, the lesser latency is expected from the IoT solution. For example, detection of fires using an IoT solution has higher criticality than detection of agricultural field parameters. The former requires a response time in the tune of milliseconds, whereas the latter can be addressed within hours or even days.

Resources: It signifies the actual capabilities of an offload location. These capabilities may be the processing power, the suite of analytical algorithms, and others. For example, it is futile and wasteful to allocate processing resources reserved for real-time multimedia processing (which are highly energy-intensive and can process and analyze huge volumes of data in a short duration) to scalar data (which can be addressed using nominal resources without wasting much energy).

Data volume: The amount of data generated by a source or sources that can be simultaneously handled by the offload location is referred to as its data volume handling capacity. Typically, for large and dense IoT deployments, the offload location should be robust enough to address the processing issues related to massive data volumes.