Chapter 3
Open Service Platforms for IoT

Preeti Agarwal and Mansaf Alam

Abstract With the advent of Internet of Things (IoT), anything on earth with embedded processor, storage, and communication technology can communicate with each other. IoT can interconnect billions and trillions of devices on earth. This is considered as next revolution in the world of internet. It is expected that this revolution will drastically improve quality of daily life, will bring new forms of collaboration, interaction, and activities. IoT is not considered a single technology, but it is an aggregation of various underlying technologies, making the application development task bit challenging. In order to cope up with this challenge, number of vendors are coming up with IoT platforms for application development. IoT platforms provide support for connecting, storing, computing, and analysing data from heterogeneous devices. This chapter, presents a reference architecture for IoT service platforms, outline a set of service and architectural requirements for IoT platform, and review four major IoT platforms (AWS IoT platform, IBM Watson platform, Microsoft Azure IoT Platform, and Google Cloud Platform) from these requirements viewpoint. Further, gaps and issues in present IoT platforms are discussed with future research directions.

Keyword Internet of Things (IoT) · IoT platform reference architecture · IoT platform service requirements · IoT platform architectural requirements

3.1 Introduction

The term "Internet of Things" was first coined by Kevin Ashton in 1999 (Ashton 2009). According to IoT concept, every sensor, every device, and every software can be connected to each other. These devices can communicate remotely with each other via. IoT platform. The main idea behind IoT was to provide ubiquitous

P. Agarwal (✉) · M. Alam
Department of Computer Science, Jamia Millia Islamia, New Delhi, India
e-mail: malam2@jmi.ac.in

M. Alam et al. (eds.), *Internet of Things (IoT)*, S.M.A.R.T. Environments,
https://doi.org/10.1007/978-3-030-37468-6_3

computing with minimum human intervention (Al-Fuqaha et al. 2015; Atzori et al. 2010). The concept gained popularity by embedding processors, storage and communication technology in devices, and these technologies converted devices into smart devices. Smart devices are now capable of sensing the environment, storing information, and can also communicate with each other, eliminating human in the loop. Enabling interaction, collaboration and communication among various devices has foster application development in many domains such as healthcare, smart homes, agriculture, traffic, and energy management (Asghari et al. 2019). In order, to provide centralized control over these smart devices, number of vendors came up with different platforms. At present, market is flooded with IoT platforms, and identifying a suitable one for a particular application is a big challenge (Agarwal and Alam 2019). Some of the popular platforms are AWS IoT, IBM Watson, Microsoft Azure, Google cloud IoT. The main objective of this chapter is:

- To provide general IoT reference architecture,
- Identify key service and architectural requirements for IoT platforms,
- Review four most popular IoT platforms from requirements view point,
- Present gaps and challenges to be addressed by future IoT platforms.

The structure of chapter is as follows: Sect. 3.2 gives general IoT reference architecture, Sect. 3.3 provides key service and architectural requirements of the platform. Section 3.4 discusses four case studies of the most popular IoT platforms AWS IoT, Microsoft Azure IoT, Google Cloud IoT, and IBM Watson IoT from requirements viewpoint. Finally, Sect. 3.5 presents challenges and open research directions to be addressed by future IoT platform.

3.2 IoT Reference Architecture

IoT is considered next revolution in World Wide Web. To provide Quality of Service (QoS), special consideration need to be taken to while defining its architecture (Fahmideh and Zowghi 2020). Many IoT architectures have been presented in literature, but the most accepted one is four layer architecture consisting of: sensor layer, gateway and network layer, service management layer, and application layer (Yaqoob et al. 2017), as shown in Fig. 3.1. Each layer is described in following subsections.

3.2.1 Sensor Layer

The first layer, called as sensor layer or perception layer. It consist of sensors and actuators, that sense the environment, collect information for processing to gain useful insights. Different kind of sensors can be deployed at this layer, like temperature, motion, humidity, sensing events etc. At this layer heterogeneous devices are deployed in plug and play manner. The perception layer digitalizes, creates secure

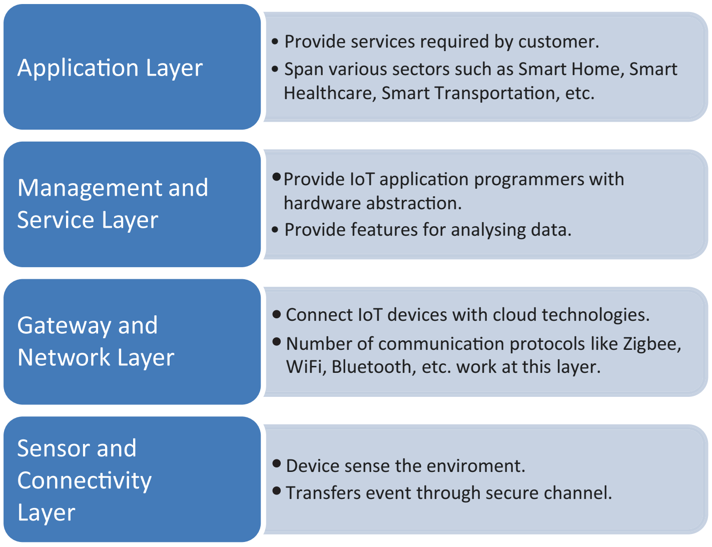
Application Layer
• Provide services required by customer.
• Span various sectors such as Smart Home, Smart Healthcare, Smart Transportation, etc.
Management and Service Layer
• Provide IoT application programmers with hardware abstraction.
• Provide features for analysing data.
Gateway and Network Layer
• Connect IoT devices with cloud technologies.
• Number of communication protocols like Zigbee, WiFi, Bluetooth, etc. work at this layer.
Sensor and Connectivity Layer
• Device sense the enviroment.
• Transfers event through secure channel.

Fig. 3.1 IoT architectural layers

channel, and transfer data to the next layer. This layer is major source of big data to be processed by next layers.

3.2.2 Gateway and Network Layer

The gateway and network layer transfers data created through secure channels by the sensor layer to the above layer. Various wireless technologies such as Zigbee, RFID, Wi-Fi, etc. are used to transmit data. Furthermore, storage and processing of data needs to be addressed at this layer. One of the possible solution for storage is cloud. Multiple cloud based storage solutions for IoT generated big data exist (Khan et al. 2017; Alam and Shakil 2016). Different ways to process both structured and unstructured data exist (Alam 2012). Most of the data management as well as resource management is carried out at this layer. Many algorithms for efficient resource management can be deployed (Ali et al. 2019).

3.2.3 Service Management Layer

Management or Middleware layer binds a service with its requestor. This layer provides features that enables the IoT application programmers to work with the sensor objects in a seamless manner, without any concern to underlying hardware. Also, this layer processes received data, make smart decisions, and based on decisions deliver the services over the network through protocols. Various analytical solutions can be applied at this layer to provide intelligent decisions.

3.2.4 Application Layer

The application layer provides the requested services to its users. For example, the application layer can provide acceleration and heart beat values to the medical care provider for its patient to continuously monitor them. This layer has ability to provide superior services to meet the user's need. The application layer provide services to many sectors such as smart home, smart healthcare, smart transportation, industrial automation and smart energy management.

3.3 IoT Platform Requirements

IoT platform is a responsible for integrating devices on network for different applications through software packages. IoT Platforms are deployed on service management layer of the IoT architecture. The platforms provide users a layer of abstraction, hiding the implementation details. IoT platforms provides an ecosystem upon which different smart applications can be built (Tiwana 2013).

Like other software's, IoT platforms also need to satisfy certain user requirements. These user requirements are divided into service requirements and architectural requirements (Razzaque et al. 2015; da Cruz et al. 2018). The service requirements can be functional and non-functional, as shown in Fig. 3.2.

3.3.1 Service Requirements

Service requirements can be classified as functional and non-functional. Functional requirements describe the required functionalities of the IoT platform, whereas non-functional requirements focus on providing QoS parameters (Razzaque et al. 2015).

The various functional requirements are described as follows:

- **Resource Discovery**: The IoT connects heterogeneous devices in dynamic environment. There must be some automated mechanism to publish, discover, and subscribe to resources in centralized manner. In order to carry out this task middleware platform maintains a registry component in which devices register themselves using necessary API's. After registration the device becomes discoverable to other devices in the network. The registered device can publish metadata about its services. Even other devices in the network can query the device by suitable query mechanism, can surf the required services and resources provided by the

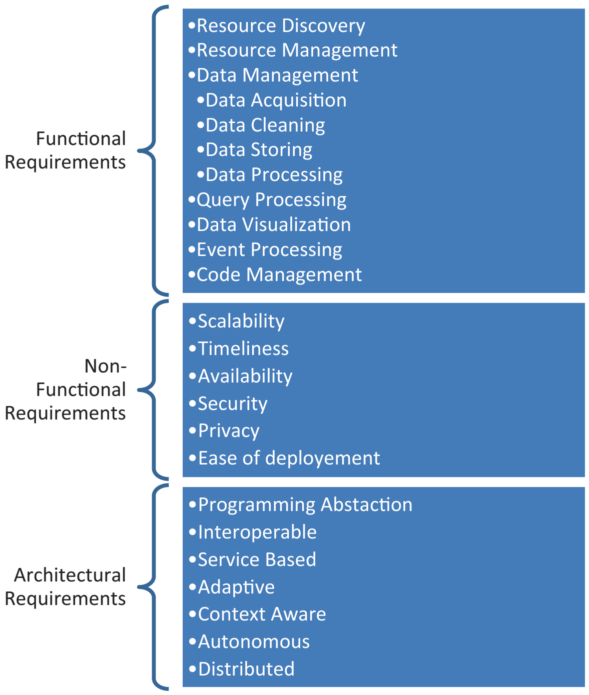

Fig. 3.2 IoT platform requirements

device. To explore services provided by a device semantic match making mechanism is used (Fabisch and Henninger 2019). The various challenges associated with discovery includes interoperability (Nitti et al. 2017), security (Zhang et al. 2017), and unique addressing (Cheng et al. 2015).

- **Resource Management:** To provide QoS, it is required to deploy certain module on IoT platform to manage resources. Resources need to be monitored, scheduled and allocated in a fair, conflict free manner (Razzaque et al. 2015) by the platform. To carry out this task platforms must maintain data about device battery time, memory usage, processing power, and other relevant information for efficient resource management (da Cruz et al. 2018).
- **Data Management**: Data management plays a vital role in any application development. In case of IoT, data refers to the data sensed by the IoT devices. IoT platform is required to provide data management services which include data acquisition, data processing, querying and visualisation.
- **Data Acquisition**: IoT platforms acquire data sensed by devices from various sources. The sensed data can be structured or semi-structured (Cheng et al. 2015). Following are the main sources of data captured by the platform (Santana et al. 2018; Khan et al. 2015):

  – Real time data about physical devices such as traffic, city maps, citizen’s data.
  – Data available in form of software such as libraries, codes, documents.
  – Historically stored data in the form of logs, historic actions.

The challenges associated with designing data acquisition are addressing, scalability, configuration, security and interoperability of objects (Apolinarski et al. 2014).

- **Data Cleaning**: This data acquired through sensors may be incomplete, inconsistent, noisy or irrelevant. To increase the reliability of the system it is required to be cleaned and preprocessed for further execution. The algorithms for anomaly detection (Cheng et al. 2015), maintaining semantic consistency, data normalization (Silva et al. 2018), and data filtering (Petrolo et al. 2014; Filipponi et al. 2010) need to be designed. Some platforms even employ strategies that allow users to collect data of interest only by filtering irrelevant data through algorithms (Soldatos et al. 2015).
- **Data Storing**: It deals with storing the data acquired through the sensors. IoT platform is mainly responsible for managing huge volume and variety of IoT sensed data. Mainly relational databases and No SQL databases are used for storage (Fahmideh and Zowghi 2020). Relational databases are used for structural, transactional data. No SQL databases are used for dynamically changing schema. Some of the popular choices of No SQL databases are Hadoop, CouchDB, HBases, and MangoDB.
- **Data Processing**: This includes the power of IoT platform to analyses sensor data for meaningful inferences. Data is usually analyzed in two modes: Real time analytics or Streaming mode, and batch processing or historical analysis (Al-Fuqaha et al. 2015). In real time analytics the concern is over timely efficient processing of fast running IoT stream data. Some of the technologies that support fast streaming data are Apache Spark, Storm. Historic analysis deals with processing batch of historically stored sensor data and finding inferences in it. Data mining techniques such as classification, regression and clustering can be done on data to find classes, groups or abnormalities, trends in acquired sensor data.
- **Query Processing**: It is the process for querying the data stored in the IoT platform. Querying mechanism also uses publish/subscribe mechanism like data acquisition (Cheng et al. 2015). The query can be a simple query or complex query aggregating data from multiple databases.
- **Data Visualization**: in response to users query on stored data, visualization techniques provide graphical view of the analysis results. Results can be in form of dashboards, maps or reports.
- **Event Processing**: The function of the platform is to process event successfully. Event is nothing, but a change in environment, which is captured by sensor, and requires certain action to be taken over it. Usually event processing is required to be carried out in real time (Cretu 2012). The platform should provide flexibility to the users in terms of executing own code and define event conditions (Sarhan 2019). The main challenges associated with event processing in IoT platform is to combine data from multiple sensor stream in different forms and combine them for one common goal. Events are usually represented in form of ontologies enabled by metadata for their correct interpretations (Fahmideh and Zowghi 2020).

- **Code Management**: IoT platform plays an important role in deploying code for IoT applications. For this code allocation, and migration services are required (da Cruz et al. 2018). Code allocation deals with selecting appropriate sensor devices and executing code on that particular sensor device. Code migration deals with providing portable facilities for migrating code on different programming services.

Non –functional requirements: The various non-functional requirements are described as follows:

- **Scalability**: The IoT platform needs to accommodate large number of devices. The IoT platform must provide features that can add any number of devices, and can remove any number of devices (Al-Fuqaha et al. 2015). Adding large number of devices must maintain QoS (da Cruz et al. 2018)
- **Timeliness**: Event processing mainly rely on timely execution on data. Most of the events require real time processing. Real time means execution of data quickly, without any delay. The time required for processing is critical and is determined by different applications processing them.
- **Availability**: For critical applications 24X7 availability of the platform is must. The platform must be available for services, even it is facing internal failures (Razzaque et al. 2015). The recovering time from failure should be very small, not affecting the system performance. Reliability and availability both are important factors and deals with fault tolerance.
- **Security and privacy**: Another important requirement of IoT platform is to provide security to the user's data. Most of the applications require to store user's personal data and information such as GPS cation, password, etc. Proper security mechanisms need to be deployed to protect user's information during transmission and storage to protect it from malicious attacks. Proper security and privacy preserving mechanisms need to be deployed at both functional and non-functional level (Al-Fuqaha et al. 2015).
- **Ease of deployment**: IoT platforms are used for end user application deployment. Most of the time they are used by application developers to integrate their own device. So, it is required from platform to be user friendly, to be easily integral with device, and must not require lot of expertise. Must be easy to install and set up.

3.3.2 Architectural Requirements

Architectural requirements supports application development. It deals with the requirements which can ease the application development task, such as programming abstraction from hardware implementation, inbuilt API's, libraries. The major architectural requirements are as follows:

- **Programming Abstraction**: The application developing programming interface of an IoT platform must be able to hide the internal working of the system. At the time of designing the IoT platform, it is required to design the level of abstraction to be provided to the different level of users.
- **Interoperable**: An IoT platform must be able to interoperate heterogeneous devices, technologies and diverse applications. These heterogeneous devices must be able to communicate with each other, exchange information, and can work collaboratively to achieve final goal. Interoperability can be achieved at network, semantic, and syntactic level (Fahmideh and Zowghi 2020).
- **Service based**: IoT platforms must support service-based framework, where new service interfaces for diverse applications can be easily added, without affecting the underlying hardware interfaces. Service oriented framework provides flexible architecture for building diverse applications.
- **Adaptive**: IoT platform architecture should be capable of adapting itself to changing environment. IoT applications usually work in dynamic environment. So, it is required that platform should incorporate this dynamicity in its architecture.
- **Context Aware**: Context awareness adds value to the information sensed by the IoT devices. Context awareness means IoT device must be capable of capturing user's information and device information for providing more meaningful inferences.
- **Autonomous**: All devices in IoT environment must work in self-governing mode. They must be able to work autonomously in collaboration with other devices in the network without human intervention (Gubbi et al. 2013; Wang et al. 2010). Automaticity can be provided by embedded intelligence, analytics, and autonomous agents (Guo et al. 2011).
- **Distributed**: In order to support distributive applications like transportation, traffic. The IoT platform must be able to provide distributed, decentralised processing. Platform must support functions that can be performed in physically distributed infrastructure environment.

3.4 Case Study of IoT Service Platforms

The market is overwhelmed with number of IoT middleware platforms. Out of hundreds of IoT platforms, this chapter presents case study of four most popular platforms namely: Amazon Web Service (AWS) IoT, Microsoft Azure IoT, Google Cloud Platform, IBM Watson IoT. Each one of the following is discussed with requirements viewpoint.

3.4.1 Amazon Web Service (AWS IoT)

AWS IoT platform is developed by Amazon Web Services (AWS IoT 2019). It can connect millions of IoT devices through secure gateway. It is an integration of large number of middleware technologies working collaboratively. AWS has lot of inbuilt technologies supporting integration with heterogeneous devices, capturing data, securely transmission on cloud, support for device authorization and authentication, analytics tools. Besides these, AWS also supports number of third party applications. The technologies supported by AWS, to satisfy service and architectural requirements of the platform are given below.

Functional Requirements AWS follows publish subscribe mechanism for resource discovery. Each device connects to AWS via. gateway through secure channel protocols. Data is then passed through Rules Engine, which transforms data and pass it to the cloud services. At cloud end, Dynamo DB for NoSQL storage is deployed, lambda functions are used for event management. AWS supports dashboards for visualisation of data.

Non-functional Requirements AWS can scale millions of devices. It interconnects devices irrespective of their underlying architecture using API's and SDK's. It supports large number of programming language, and supports code migration. It can integrate with large number of Operating system through command line interface. Each device is authenticated and authorised using certificate, and is assigned unique ID. Data during transmission is secured through encryption and decryption. Overall AWS is easy to deploy, and uses pay as you use policy.

Architectural Requirements AWS provides programming abstraction, as user can program in any language. AWS has support for large number of interfaces. It has high fault tolerance and availability. Things Shadow maintains data related to device such as identification, location. It has centralized architecture, but supports distributed processing with Elastic MapReduce. The different features corresponding to requirements are summarized in Table 3.1.

3.4.2 Microsoft Azure IoT

Microsoft Azure IoT is developed by Microsoft (Azure IoT | Microsoft Azure 2019). It is considered to be only hybrid cloud service solution. Unlike, AWS IoT it can support pre-configured solutions. Azure has one of the powerful artificial intelligence support engine. It has number tools and technologies for securing capturing data, storing, and processing data. In response to various service and architectural requirements, various tools and technologies deployed are discussed below.

Table 3.1 AWS IoT features

Service requirements	
Functional requirements	
Resource discovery	Publish and subscribe mechanism via. Message broker on device gateway through MQTT or HTTP protocol
Resource management	Things registry is used to manage resources allocated with each device. Elastic load balancing to manage load in network
Data management	Amazon simple storage service (S3) provides scalable storage, dynamo DB provides NoSql databases, lambda for virtualization
Query processing	Rules engine using SQL language for message processing
Data visualization	Dashboards for visualization
Event processing	Amazon Kenesis for real time event processing. Simple notification service is used for event notifications. Lambda functions can trigger different events
Code management	Through in built SDK's can code in any language
Non-functional requirements	
Scalability	Can scale billions of heterogeneous devices
Timeliness	Amazon Kenesis with Kenesis analytics is used for real time processing
Availability	24 × 7
Security and privacy	Certificates for authentication, supports encryption decryption of data, user can define its own security rules and policies
Ease of deployment	Easy. With just registration
Architectural requirements	
Programming abstraction	Command line Interface is compatible with number of OS such as windows, Linux, and OSX. AWS SDK's allow user to program in any language
Interoperable	Rules engine makes it interoperable with other services
Service based	Support for large number of API's and SDK's make architecture service based
Adaptive	Not very adaptive friendly platform
Context-Awareneness	Things shadow maintain current state information of device
Autonomous	Each device can autonomously collaborate with other devices to exchange data using its unique identification number
Distributed	It has centralized architecture, but uses elastic MapReduce for processing

Functional Requirements Each device registers itself with azure directory service, which provides it a unique identification. Load balancing by azure platform is carried out both at local as well as global level. Supports both NoSQL storage for semi structured data, as well as support for SQL storage. It supports large azure analytics engine. Number of powerful tools for visualisation are also available. Supports real time event processing and code management.

Non-functional Requirements Azure can be scaled to large number of devices. Supports both real time as well as historic processing of data. In order to improve availability, back up mechanism is deployed to make system fault tolerant. To provide device authentication, two way secure protocol is deployed.

Architectural Requirements Azure can be deployed either in windows, or linux platform. It has support for number of programming languages such as python, java, PHP, Node.js, etc. To interconnect large number of heterogeneous devices with different underlying architecture, number of inbuilt SDK's and APIs are available. Mean Time to Failure is quite low. Supports decentralized serverless architecture.

The service and architectural requirements of Microsoft azure are summarized in Table 3.2.

3.4.3 Google Cloud Platform

Google Cloud Platform (GCP) is developed by Google. It can connect heterogeneous IoT devices. It can support lightweight applications. It is considered to be server less architecture. GCP has support for large number of analytics libraries like tensor flow, etc. GCP is one of the powerful emerging IoT platform supporting large number of features, including support mobile applications. Various features supporting functional, non-functional and architectural requirements are given below:

Functional Requirements GCP uses publish subscribe mechanism for resource discovery. A registry of device is maintained. It supports BigTable storage solution, and firebase solution for real time processing of the event. Supports large number of programming languages interface.

Non-functional Requirements Can scale millions of devices. In order to make system fault tolerant, it deploys back up mechanism. Security and privacy is provided by authorisation and authentication of device. Firebase solution deals with the timeliness of the event execution. It supports large number of machine libraries like tensorflow to provide intelligent solutions.

Architectural Requirements GCP provides good programming abstraction by supporting large number of programming languages. Context aware applications are easy to deploy as each device in the network can be tracked by unique ID, and location. Suitable for lightweight mobile apps as well as client web applications. It uses centralized architectural approach.

The Service and architectural requirements of Google Cloud Platform are summarized in Table 3.3.

Table 3.2 Microsoft Azure features

Service requirements	
Functional requirements	
Resource discovery	Not much support for resource discovery
Resource management	Azure active directory. Supports both global level and local level load balancing
Data management	Support storage using Cosmos DB and processing, support for in-motion analytics. Data can also be stored in blob storage and Postgre storage
Query processing	Azure analytics through SQL syntax
Data visualization	Dashboards, power BI, web apps
Event processing	Support for predictive analytics, user can set a thresholds and alert limits through event hub
Code management	Supports code migration
Non-functional requirements	
Scalability	Scalability of ten million devices per instance
Timeliness	Support both historic and real time processing. Supports stream analytics
Availability	24 × 7. supports fault localization. Maintains backup to improve reliability
Security and privacy	Device authentication through two way secure protocol
Ease of deployment	Medium level difficulty
Architectural requirements	
Programming abstraction	Can scale with number of programming languages and command line interface through inbuilt tools and technologies
Interoperable	Agent libraries, SDK's allow interoperability within heterogeneous systems
Service based	Provides certain pre- configured solutions. Support number of architectures like event driven, microservices, n-tier, web-queue, and worker
Adaptive	MTTR is very low
Context-awareness	Azure active directory
Autonomous	Heterogeneous devices can work autonomously in collaboration to each other
Distributed	Decentralized serverless architecture

3.4.4 IBM Watson IoT

IBM Bluemix platform is recently named as IBM Watson IoT platform. It supports large number of cognitive analytics, and considered one of the popular platform for researchers (IBM Knowledge Center 2019). The various technologies provided by IBM Watson in view of functional, non-functional and architectural requirements are as follows:

Functional Requirements Each device in the network is connected via. Message broker through secure gateway. Each device is assigned a unique organisation ID. It supports both SQL databases and NoSql databases. Cognitive engine is deployed

Table 3.3 Google Cloud platform features

Service requirements	
Functional requirements	
Resource discovery	Uses pub/ sub scheme. Devices register through device id. Maintains registry of devices
Resource management	Devices managed through device drivers and protocol bridge. Deploy resource allocation
Data management	Cloud BigTable is used for data storage. Firebase database for real time processing. Supports both historic and real time data processing
Query processing	BigQuery is use for query processing
Data visualization	Dashboards
Event processing	Firebase database for real time event time processing
Code management	Support number of programming languages and libraries
Non-functional requirements	
Scalability	Can scale millions of devices
Timeliness	Real time processing though firebase
Availability	24 × 7. Fault tolerance through backup
Security and privacy	Authentication, authorization of device
Ease of deployment	Easy
Architectural requirements	
Programming abstraction	Number of programming languages
Interoperable	Number of libraries, lightweight libraries, APIs, SDK
Service based	Supports mobile apps, and end to end applications
Adaptive	Not very adaptive
Context-Awareneness	With each device there is unique id, state information, telemetry
Autonomous	Each device with unique id can send stream of telemetry
Distributed	Centralized approach

for query processing. Dashboards and Jupyter notebook for visualization. Supports both historic and real time event processing.

Non-Functional Requirements can scale millions of devices. Kafka REST APIs are used to provide timeliness of event processing. Provide authentication, authorisation mechanism through protocols, with support for risk management. Platform is quite easy to deploy, making it popular choice among researchers.

Architectural Requirements Supports large number of programming language. Usually Node Red editor is used for programming. Suitable for deploying context aware apps. Supports large number of SDKs. Can be used to build lightweight as well as web apps. It has centralized architecture with support for distributed processing. The service and architectural requirements for IBM Watson are summarized in Table 3.4.

Table 3.4 IBM Watson features

Service requirements	
Functional requirements	
Resource discovery	Message broker for secure device registration via. Gateway
Resource management	Each device registers through unique organization id
Data management	Cloudant NoSql DB for real time processing. Data lake for SQL storage. DB2 for long term schema storage
Query processing	Has cognitive engine for providing data analytics capability. Supports machine learning. Supports both predictive and prescriptive analytics
Data visualization	Dashboards, support jupyter notebook
Event processing	Support historic and real time event processing
Code management	Supports code migration
Non-functional requirements	
Scalability	Can scale millions of devices
Timeliness	Real time processing through no SQL event streaming and Kafka REST APIs
Availability	High availability
Security and privacy	Unique authentication code for device identification. Supports risk management. Secure communication via. Protocols
Ease of deployment	Easier to deploy
Architectural requirements	
Programming abstraction	User can choose own programming language and architecture. Red node visual programming editor
Interoperable	Large number of SDK's
Service based	Framework supports client side application, mobile micro apps
Adaptive	Can adapt to changing environment
Context-awareneness	Stores device id, last activity, geographic location
Autonomous	Each device has unique social id and can autonomously collaborate with each other
Distributed	Centralized architecture with support for distributed processing

3.5 Challenges and Open Research Problems

The IoT Platforms, still have lot of research challenges that need to be addressed in future research. The various challenges and open research problems related to IoT platforms are discussed below:

- Need for improved and more accurate models for resource discovery. IoT platform needs to address large number of request in timely and accurate manner. The present registry methods such as distributed, hybrid, or probabilistic does not fully cater this requirement (Teixeira et al. 2011). So, there is a need for designing better resource discovery models.

- Need for more efficient resource scheduling and management policies. Conflict of resources among IoT devices is a very common scenario. At present very few IoT platforms deploy strategies for resource conflict resolution. So, this is one of the area which need to be addressed in the future.
- More focus on support for data filtering. Data aggregation and filtering is one of the major steps in analysis. Most of the platforms provide data aggregation solution, but very few support on filtering of relevant and irrelevant data.
- Support for data compression. IoT generates huge amount of data, requiring large amount of storage. Proper data compression solutions can be deployed to reduce the amount of storage required.
- Support for changing business logic or IoT environment. Business logic and requirements keep on changing with time. Proper code allocation and migration strategies need to be deployed with support for firmware update.
- Support for Hard real time processing is required. Soft real time processing of events is done efficiently IoT. But, still addressing Hard time event is a challenge.
- Seamless replacement of modules to provide reliability in case of faults and failures is still a challenge.
- Service provising in case of failure to improve availability of a platform in seamless manner needs to be addressed in future research.
- Human intervention is required for deployment of devices on IoT platform. There is a need for pre-configured solutions to address this issue and reduce human efforts, making things automated and ease for deployment.
- Need to address syntactic and semantic interoperability. Most of the IoT platforms provide hardware interoperability with lesser focus on syntactic and semantic interoperability.
- Need for more dynamic rules and policies for making deployed system adaptable to run time environment.

3.6 Conclusion

In IoT scenario heterogeneous devices are connected and work collaboratively to achieve a particular goal. At the lowest layer, lies the resource constrained sensors, that senses the environment. To convert this information into meaningful insights, the data through sensors need to be stored and processed. All these packages need to be consolidated into one platform called as IoT platform. Number of vendors are available in the market with different requirements and specifications. Choosing the right one according to the need of application is the major key factor in successful deployment of application. The IoT platforms have come a long way, still they need to work on certain aspects to provide better functionality like security, resource provision, making more user friendly, easy device discovery, deployment and integration, better storage and resource management policies.

References

Agarwal, P., & Alam, M. (2019). Investigating IoT middleware platforms for smart application development. *arXiv preprint arXiv, 1810*, 12292.

Alam, M. (2012, December). Cloud Algebra for handling unstructured data in cloud database management system. *International Journal on Cloud Computing: Services and Architecture (IJCCSA)*, *2*(6) ISSN: 2231 – 5853 [Online], 2231 – 6663 [Print], https://doi.org/10.5121/ijccsa.2012.2603, Taiwan.

Alam, M., & Shakil, K. A. Presented "Big Data analytics in Cloud environment using Hadoop. In International conferences on Mathematics, Physics & Allied sciences-2016, March 03–05, 2016, Goa.

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials, 17*(4), 2347–2376.

Ali, S. A., Affan, M., & Alam, M. (2019). A study of efficient energy management techniques for Cloud Computing environment. *2019 9th International conference on Cloud Computing, Data Science & Engineering (Confluence)* (pp. 13–18), Noida, India. https://doi.org/10.1109/CONFLUENCE.2019.8776977.

Apolinarski, W., Iqbal, U., & Parreira, J. X. (2014, March). The GAMBAS middleware and SDK for smart city applications. In *2014 IEEE International conference on pervasive computing and communication workshops (PERCOM WORKSHOPS)* (pp. 117–122). IEEE.

Asghari, P., Rahmani, A. M., & Javadi, H. H. S. (2019). Internet of things applications: A systematic review. *Computer Networks, 148*, 241–261.

Ashton, K. (2009). That 'internet of things' thing. *RFID Journal, 22*(7), 97–114.

Atzori, L., Iera, A., & Morabito, G. (2010). The internet of things: A survey. *Computer Networks, 54*(15), 2787–2805.

AWS IoT. (2019). Retrieved September 12, 2019, from https://docs.aws.amazon.com/iot/index.html

Azure IoT | Microsoft Azure. (2019). Retrieved September 12, 2019, from https://azure.microsoft.com/en-in/overview/iot/

Cheng, B., Longo, S., Cirillo, F., Bauer, M., & Kovacs, E. (2015, June). Building a big data platform for smart cities: Experience and lessons from santander. In *2015 IEEE International congress on Big Data* (pp. 592–599). IEEE.

Cretu, L. G. (2012). Smart cities design using event-driven paradigm and semantic web. *Informatica Economica, 16*(4), 57.

da Cruz, M. A., Rodrigues, J. J. P., Al-Muhtadi, J., Korotaev, V. V., & de Albuquerque, V. H. C. (2018). A reference model for internet of things middleware. *IEEE Internet of Things Journal, 5*(2), 871–883.

Fabisch, M., & Henninger, S. (2019). ESPRESSO–systemic standardisation approach to empower smart cities and communities. *Smart Cities in Smart Regions, 2018*, 115.

Fahmideh, M., & Zowghi, D. (2020). An exploration of IoT platform development. *Information Systems, 87*, 101409.

Filipponi, L., Vitaletti, A., Landi, G., Memeo, V., Laura, G., & Pucci, P. (2010, July). Smart city: An event driven architecture for monitoring public spaces with heterogeneous sensors. In *2010 Fourth International Conference on Sensor Technologies and Applications* (pp. 281–286). IEEE.

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems, 29*(7), 1645–1660.

Guo, B., Zhang, D., & Wang, Z. (2011, October). Living with internet of things: The emergence of embedded intelligence. In *2011 International conference on internet of things and 4th international conference on cyber, physical and social computing* (pp. 297–304). IEEE.

IBM Knowledge Center. (2019). Retrieved September 12, 2019, from https://www.ibm.com/suport/knowledgcenter/SSQP8H/iot/overview/architecture.html

Khan, Z., Anjum, A., Soomro, K., & Tahir, M. A. (2015). Towards cloud based big data analytics for smart future cities. *Journal of Cloud Computing, 4*(1), 2.

Khan, S., Shakil, K. A, & Alam, M. (2017). *Cloud based Big Data Analytics: A survey of current research and future directions, Big Data Analytics* (pp 629–640). Springer, Print ISBN: 978-981-10-6619-1, Electronic ISBN: 978–981-10- 6620-7.

Nitti, M., Pilloni, V., Giusto, D., & Popescu, V. (2017). Iot architecture for a sustainable tourism application in a smart city environment. *Mobile Information Systems, 2017*.

Overview of Internet of Things | Solutions | Google Cloud. (2019). Retrieved September 12, 2019, from https://cloud.google.com/solutions/iot-overview

Petrolo, R., Loscri, V., & Mitton, N. (2014, August). Towards a smart city based on cloud of things. In *Proceedings of the 2014 ACM international workshop on wireless and mobile technologies for smart cities* (pp. 61–66). ACM.

Razzaque, M. A., Milojevic-Jevric, M., Palade, A., & Clarke, S. (2015). Middleware for internet of things: A survey. *IEEE Internet of Things Journal, 3*(1), 70–95.

Santana, E. F. Z., Chaves, A. P., Gerosa, M. A., Kon, F., & Milojicic, D. S. (2018). Software platforms for smart cities: Concepts, requirements, challenges, and a unified reference architecture. *ACM Computing Surveys (CSUR), 50*(6), 78.

Sarhan, A. (2019). Cloud-based IoT platform: Challenges and applied solutions. In *Harnessing the Internet of Everything (IoE) for accelerated innovation opportunities* (pp. 116–147). Hershey: IGI Global.

Silva, B. N., Khan, M., & Han, K. (2018). Internet of things: A comprehensive review of enabling technologies, architecture, and challenges. *IETE Technical Review, 35*(2), 205–220.

Soldatos, J., Kefalakis, N., Hauswirth, M., Serrano, M., Calbimonte, J. P., Riahi, M., et al. (2015). Openiot: Open source internet-of-things in the cloud. In *Interoperability and open-source solutions for the internet of things* (pp. 13–25). Cham: Springer.

Teixeira, T., Hachem, S., Issarny, V., & Georgantas, N. (2011, October). Service oriented middleware for the internet of things: A perspective. In *European conference on a service-based internet* (pp. 220–229). Berlin/Heidelberg: Springer.

Tiwana, A. (2013). *Platform ecosystems: Aligning architecture, governance, and strategy*. Oxford: Newnes.

Wang, H., Zhou, X., Zhou, X., Liu, W., Li, W., & Bouguettaya, A. (2010, December). Adaptive service composition based on reinforcement learning. In *International conference on service-oriented computing* (pp. 92–107). Berlin/Heidelberg: Springer.

Yaqoob, I., Ahmed, E., Hashem, I. A. T., Ahmed, A. I. A., Gani, A., Imran, M., & Guizani, M. (2017). Internet of things architecture: Recent advances, taxonomy, requirements, and open challenges. *IEEE Wireless Communications, 24*(3), 10–16.

Zhang, K., Ni, J., Yang, K., Liang, X., Ren, J., & Shen, X. S. (2017). Security and privacy in smart city applications: Challenges and solutions. *IEEE Communications Magazine, 55*(1), 122–129.