Chapter 4: Architecture for an Enterprise IoT

4.1 Introduction

The Internet of Things (IoT) creates a disruptive transformation across several areas. Since being invented, the IoT applications have been significantly found in consumer oriented market segments like healthcare and home automation. In parallel bigger companies of the world started exploring the applications of Internet of Things in other areas like agriculture, governance of cities, manufacturing, health care etc.

Industrial IoT (IIoT), specifically focuses on the requirements of industrial applications, such as manufacturing, supply chain management, industrial engineering, oil and gas and utilities. In a larger picture, we may not be able to find much difference in the implementation of Industrial IoT solutions from customer oriented solutions, since the technical components (both hardware and software) involved in the end to end solutions are almost the same (with respect to the sensors, cloud platforms, connectivity and analytics). However, there are several very important aspects that distinguish Enterprise IoT from the other application areas. In this chapter, we try to address those differences which are usually seen from the perspectives of business owners, vendors, delivery managers, customers and other stakeholders of Enterprise IoT Applications.

4.2 IoT Architecture for an Enterprise

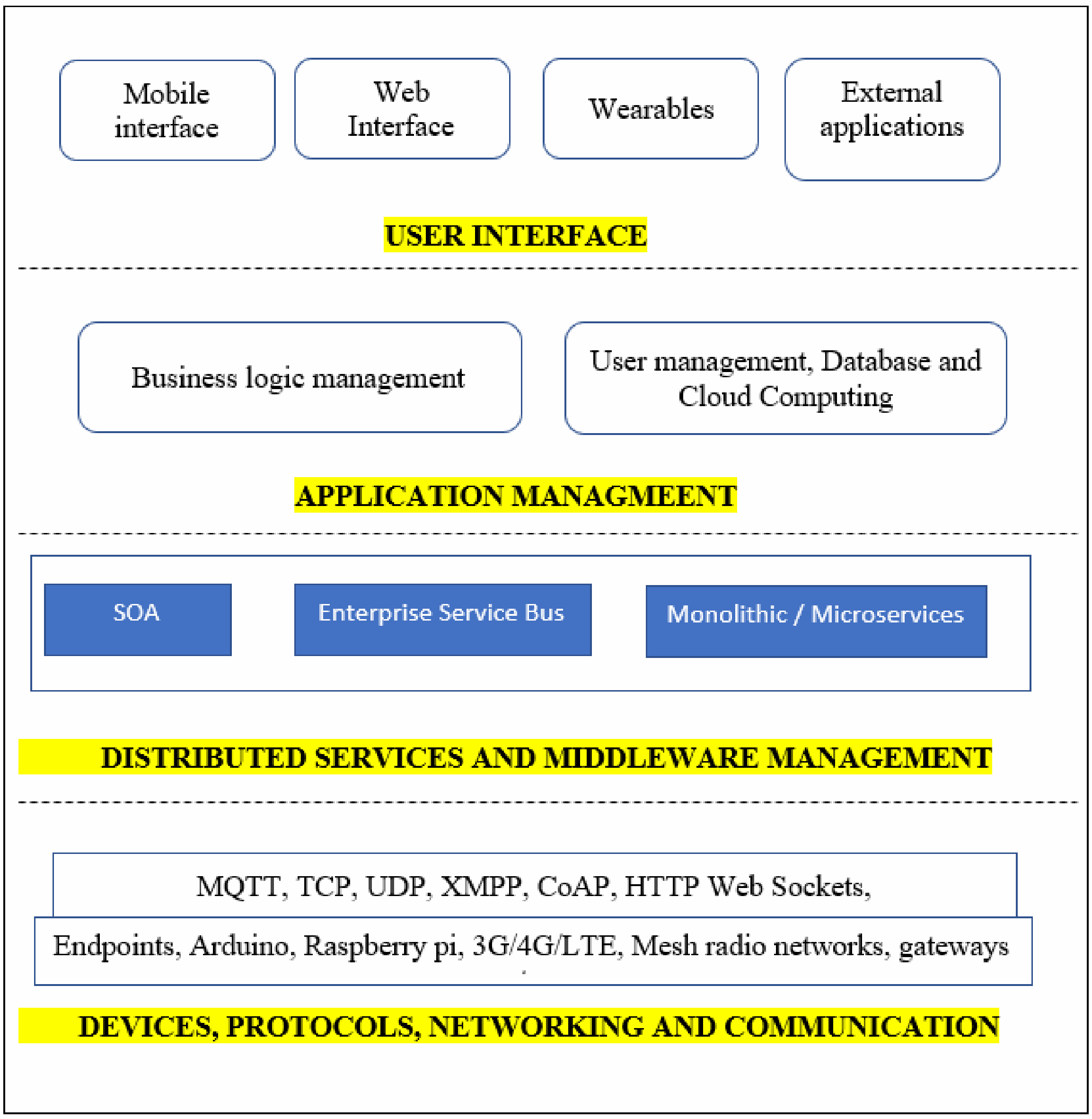
Internet of Things cannot be defined in a single Architectural model due to the participation of various hardware, software and protocols or standards. A modular and scalable architecture with a variety of capabilities will be required to understand the complete functionalities of Internet of Things. In Enterprise Internet of Things it gets more complicated due to the expectations from various types of end users and huge

amount of data with different formats. Moreover, we need to cover multiple components like server side entities, cloud, networking model that communicate with the devices, agents, edge, middleware etc., However, we are trying to provide a reference architecture which can be a seen as a combination of various entities that enable an Internet of Things application in an Enterprise.
This architecture is for generic reference and it does not specify or emphasize any technology family. The architecture can be extended based on the future advancements in the technology and industrial requirements.

Basically, the reference architecture splits the various components involved into multiple layers. Each layer is dedicated for a specific set of functionalities performed by hardware and software entities. Each layer communicates with the bottom and upper layers using specific communication methodologies and protocols. Collectively the architecture ensures that functional and non-functional requirements of an Enterprise IoT system are satisfied by developing a solution based on the architecture. There are also some entities or functionalities which span across the layers. We present this architecture as simple as possible to enable the readers to understand the basics.

Based on the reference architecture, the various components of an Enterprise IoT system can be divided into following layers.

1. User interface and External communication layer
2. Application management layer with event processing, storage and higher order analytics
3. Enterprise service bus or middleware layer
4. Communication and networking layer
5. Device layer



Mobile interface
Web Interface
Wearables
External applications
USER INTERFACE
Business logic management
User management, Database and Cloud Computing
APPLICATION MANAGMEENT
SOA
Enterprise Service Bus
Monolithic / Microservices
DISTRIBUTED SERVICES AND MIDDLEWARE MANAGEMENT
MQTT, TCP, UDP, XMPP, CoAP, HTTP Web Sockets,
Endpoints, Arduino, Raspberry pi, 3G/4G/LTE, Mesh radio networks, gateways
DEVICES, PROTOCOLS, NETWORKING AND COMMUNICATION

Fig. 4.1 A Reference Architecture for IoT

Access management, Security and User management are the cross-cutting layer activities performed across the layers.

Approaching Enterprise IoT with the support of a Reference architecture is important. As the IoT technology has become a recognized way of monitoring the events, several industrial environments and enterprises started implementing it. The data produced by sensors and objects are becoming huge in larger environments and the task is massive to handle the data from all the sensors according to the already complicated business processes. In Enterprises IoT services should coordinate technology, human and business process to quickly interact with the real world across the business domains. Due to the presence of variety of business processes, personal and physical entities, and the changes in physical world the Enterprise IoT is several times complicated to implement.

With the development of technology and necessity to make it useful in enterprises, the IoT application developers have started to consider how to integrate the existing network facilities and the new opportunities in system design. The huge information collected as an outcome of connected environment is further processed and delivered to different applications according to the requirements, then is used to trigger the corresponding collaborative business system. Thus, it would not going to be a single independent technology challenge, but a challenge to be handled in technology, business and management layers of the Enterprise.

Comparatively Enterprise IoT brings new technical challenges to IoT services such as, heterogeneity of hardware, network and operating system, interoperability between applications and services, fusion of massive heterogeneous data, scalability and continuous integration. Available IoT devices (sensors and actuators) are vendors specific where the communication protocols and data exchange formats vary from device to device. The complexity increases exponentially when simple devices are integrated to form a complex network and it is difficult to provide a unified solution for IoT application. In the IoT application, many actors comprising human and nonhuman objects and many systems are designed for specific applications, adopting specific data standards and communication platforms. Different platforms lead to great inconvenience for achieving interoperability and mutual communication in IoT applications. Most of the challenges concerned with communication among different layers of Internet of Things is usually handled by Middleware layers. Middleware layers provide necessary abstraction to the device layers and it enables deployment of various methodologies in the higher layers to overcome the above challenges stated.

Device Layer

Any device can be a part of Internet of things if it is provided with a communication mechanism so that it can exchange data with the internet. If they do not have the communication ability naturally, they have to be used with micro controller devices which has direct connectivity to the internet with the help of Wi-Fi or Blue tooth modules attached with them. Some of the devices with direct connections are

- Arduino with Ethernet, Wifi, or GSM
- Raspberry pi with Ethernet, Wifi or GSM
- Intel Galileo

Also, several devices following standards like ZigBee, low power radio devices etc., also can be connected to the internet and they can become a part of the IoT solution.

Each device has to be given with an identity, so that it can be used as an address for communication. The identity can be a chip level unique identifier or provided by the communication sub system such as Bluetooth label or MAC address in Wi-Fi. In some cases the identify is available in the EEPROM itself. The reference architecture recommends that an immutable, permanent UUID has to be provided to each device.

Communication and Networking layer.

The communication layer is responsible for connecting all the devices with each other and to the primary network backbone and to the internet. In this process, we have several communication protocols and the following are few of them.

- HTTP/HTTPS

  A simple stateless text-based protocol which is used as a request reply protocol in the application layer and the small devices like 8 bit controllers hardly support it fully. POST and GET are the primary methods of this protocol and to utilize all the capabilities of it the devices need to be 32 bit based.
- MQTT 3.1/3.1.1

  MQTT is the protocol optimized for Internet of Things. It was invented to support embedded system and SCADA. This protocol is a publish-subscribe messaging system based on a broker architecture model. This protocol has been designed to function with weaker networks where data loss and connection loss in between are frequent. It has been designed to function over TCP. MQTT has better adoption and wider library support when compared to CoAP.

- Constrained application protocol (CoAP)

  CoAP also is a protocol optimized for Internet of things and this is from the IETF. It supports RESTful services and works based on client server approach. It can be used over UDP. The connectivity over firewalls and NAT networks are not simple when compared to the MQTT.

Both MQTT and CoAP have their own strengths and weaknesses and they are used according to the situations.

Enterprise service bus or middleware layer

There is a need for aggregation layer to consolidate and enable the communication among the brokers and other entities. This is the primary service bus, something like a backbone where everything is connected to each other. It is very significant in the architecture since it supports an HTTP server and MQTT broker to communicate with the devices. This layer has gateways to combine the communications from different devices and route them to appropriate destinations. They function as bridge among the different protocols for making communication possible from end to end. For instance, it supports HTTP based APIs and MQTT messages. Along with that, it also provides supports to the legacy protocols. It also performs critical role in ensuring the security by enforcing the policies. Depends upon the Enterprise requirements the identity and access management will have various complexities and the middleware layer supports them to function.

Application management layer with event processing, storage and higher order analytics

On top of Enterprise service bus, the application management layer takes care of higher level application concerns. It should have the capability to handle data into databases and to process the events when they occur in the system. The application layer part of the Enterprise Architecture is the most versatile part where the developers have various technology choices to build their applications and databases. In most of the places, the amount of data and arrival of data is huge and they prefer support of a separate big data analytics platform. A cloud scalable platform lie Apache Hadoop provides very effective analytics such as map reduce. Event processing to manage the real time functionalities of the system also taken care by the application programs.

User interface and External communication layer

The Enterprise application should provide well defined methods to communicate with the outside world. The external world includes various types of users with different access methods like smart phone application, web application, a kiosk or similar interface and external applications like web services which would communication with the enterprise application for integration or auditing purposes. Web based front ends are highly popular due to their access in any device with an internet connectivity. User groups can be created with different access permissions to work with the web based dash boards where the application programs render various analytics and information.

Any enterprise application can be made to talk with another application with the help of APIs. API management system also available in this layer which manages and controls the APIs. The API has to be continuously updated with newer versions and they have to be made available to the users. Several technologies such as Python, Ruby, PHP, Servlets/JSP are available to build the applications in the application tier. Smart phone applications are also getting popular due to their portability and availability of additional sensors.

4.3 Middleware solutions for Enterprise IoT

Internet of Things Middleware is software that functions as an interface between components of the IoT, making communication possible among disparate elements that would not be naturally speaking with each other. IoT middleware deals with the structure, format and encoding of the information that is being exchanged between different layers, devices and sensors. It will act as a connecting bridge and common standard amongst the diversity of devices, sensors, OS and applications that will make up the IoT ecosystem architecture. Middleware is needed for integration before data can be analyzed.

Independent disconnected systems will create siloes of IoT ecosystems which will impact the agility and scalability of the whole system development. So the middleware design is very critical in end to end solution development. The functional components of the IoT middleware is illustrated in figure 1.

Middlware technologies are not just serving IoT applications. They are known for interconnecting different platforms from very early stages of software development. Beginning from Common Request Broker Architecture (CORBA), COM, DCOM there were several middlware solutions functioning in Enterprise Application Integration (EAI) space. The recent developments were web services which has made development and integration of Enterprise solutions without any dependency of platforms or languages or other technologies.

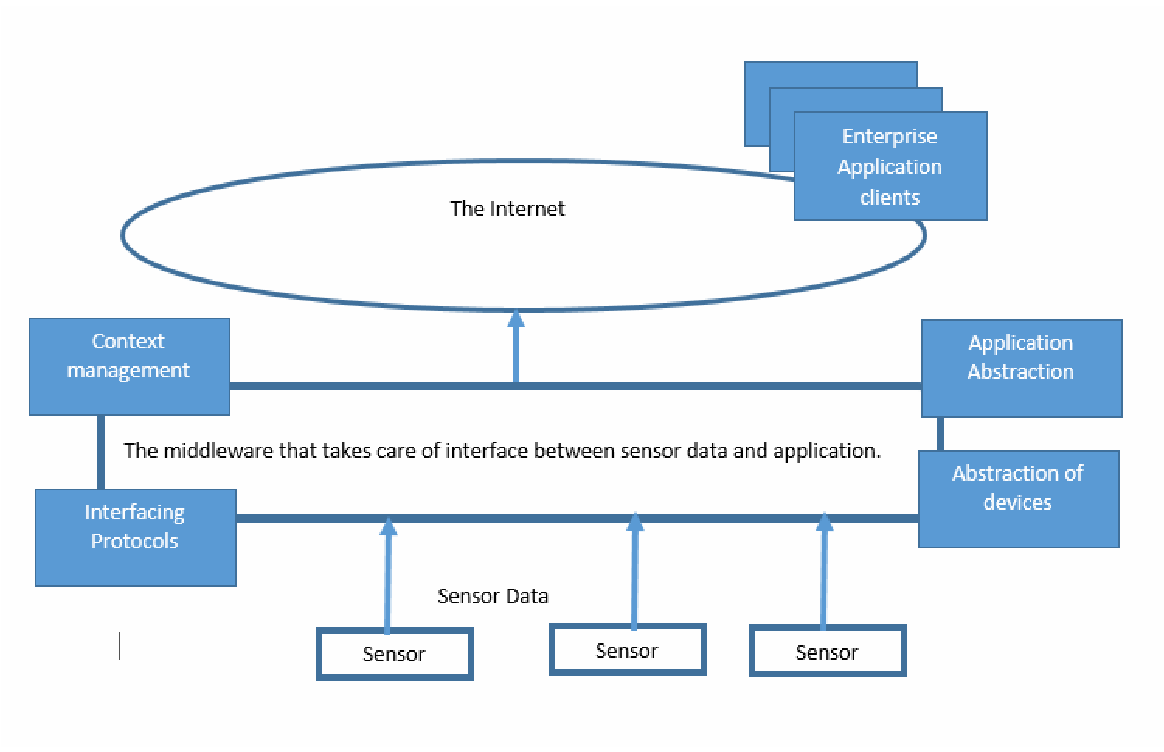


Figure: 4.2 Middleware functional components for Enterprise IoT

The middleware plays a crucial role in development of application. API management is a critical function of middleware. Developers use to see everything in terms of programmatic calls to methods. It may be a physical device or a protocol or any interface, the middleware enables handling them in terms of simple API management.

The middleware tries to call APIs from disparate systems, taking care of interconnections. So it needs to be scalable and configurable. The middleware should support dynamic changes in the number of devices, amount of data or any higher level business requirements such as user managements and permissions. New sensor or device should automatically be inducted into the ecosystem

Management of APIs along with other functions like routing, transformation and messaging also critical challenges to be solved by middleware designers. Several proprietary middleware solutions are available now to build the Enterprise IoT systems. Mulesoft, Oracle, RedHat and WSO2 are among the companies that offer IoT middleware.

These products provide API management as well as basic messaging, routing and message transformation. More comprehensive IoT platforms include middleware along with sensors and networking components. The services provided by these middleware are different according to their specific scope and the Enterprises prefer to choose one of them according to their requirements.

4.4 Monolithic Software Architectures

With the continuous development and evolvement of Internet of Things (IoT), Enterprise IoT applications become much larger in scale and even more complex in structure. This leads to poor scalability, extensibility and maintainability.

In the monolithic architecture software system is deployed as a single solution, in which functionally distinguishable aspects are all interwoven. The natural advantages of monolithic architecture are module independent, uniform standards and technology stack. Many researches offer solutions from perspective of monolithic architecture. Earlier IoT studies mainly focus on hardware network and low-level software technology.

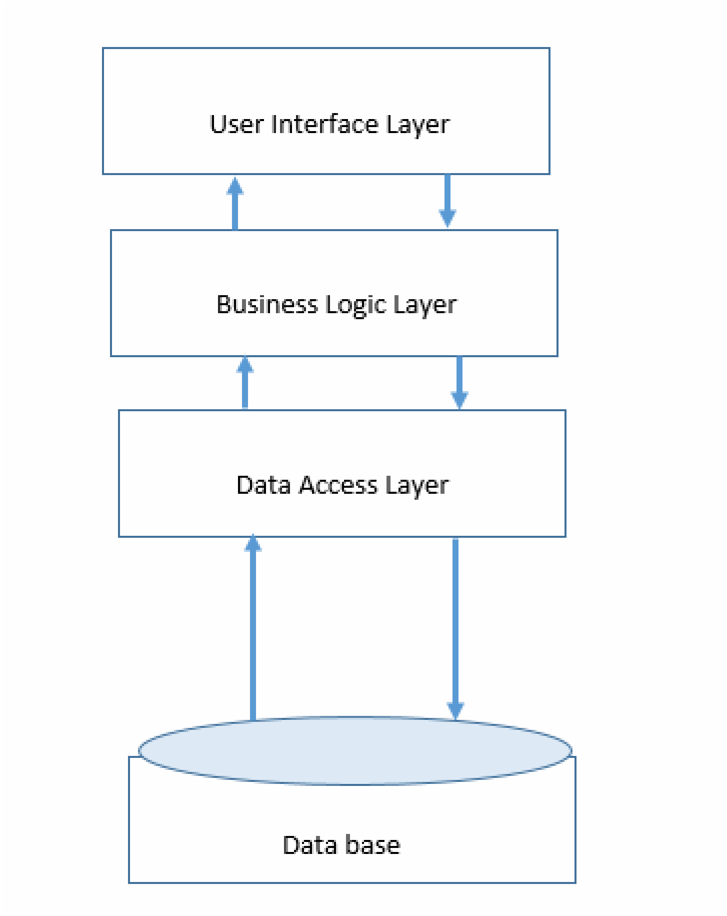


Figure 4.3 : Monolithic architecture of multi-tier application model

Monolithic architecture has some inevitable defects. First, the entire system is a united application; only multiple deployments can improve the system performance, while the overloaded functions create bottleneck, which is a waste of computing resources. Second, in the case of the change and evolution of the system, a change in a function may affect other functions due to high dependencies. This also brings complexity for re-deployment, maintenance and continuous integration. Finally, the whole system uses a sole technology stack and development standards, which in turn limits the methods to solve the problem of physical heterogeneity.

4.5 Micro Services Applications

Microservices describes a software architecture built around small, decoupled processes that communicate through language-agnostic APIs. The Microservices landscape is being influenced by the impact of mobile and Internet of Things

(IoT), which increase the pressures to improve and accelerate software delivery. Microservices approach is a way of breaking down applications into manageable units of executables.

Instead of in-memory function calls, separate processes will be implemented. These are functionally atomic and highly decoupled and task specific. They communicate with each other using lightweight mechanisms like REST API.

Several prominent corporate have migrated to Microservices. Netflix, eBay, Amazon, Twitter, PayPal, Soundcloud are few of them. They have experienced immense benefits in terms of maintainability of their system and continuous development and integration.

In the field of IoT applications, Microservice architecture has been introduced in response to the challenges faced by monolithic applications. The primary merits considered were its flexibility, lightweight and loose coupling. It has better capacity to support interoperability and accommodate heterogeneous objects. In addition, this framework can easily achieve more application integration such as automation, intelligence, Geo service and Big Data.

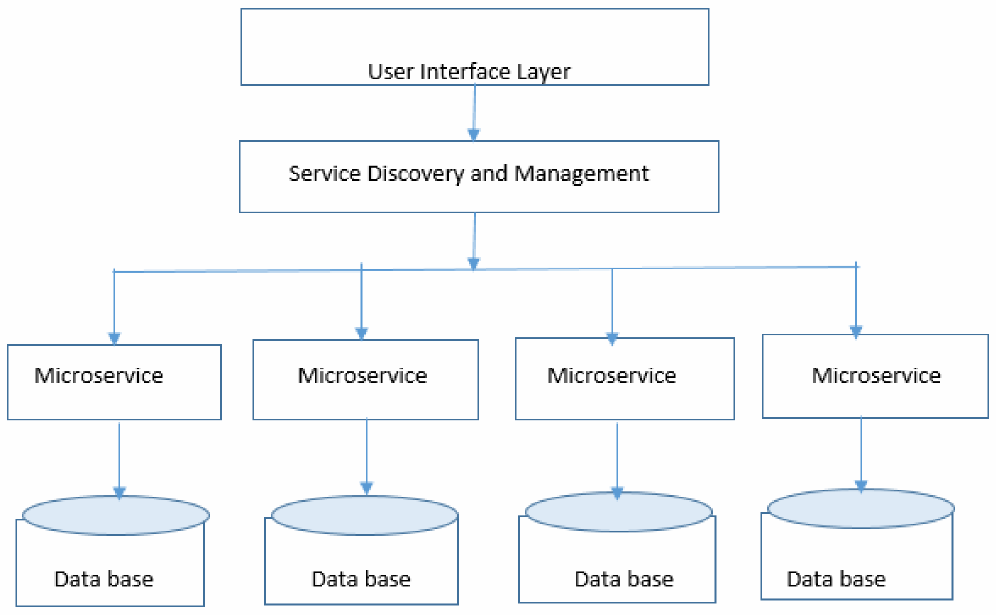


Figure 4.4 Microservices model of an application

Micro service architecture is considered as a new software design pattern. It suggests that a single large and complex application should be divided into small manageable groups, where each group deals with the related services. According to its specific business responsibilities, each micro service is dedicated for a single business function. Any applicable tools and languages can quickly realize a specific service. Compared with the traditional monolithic architecture, the Micro service architecture has obvious advantages, such as complexity under control, independent deployment, more choices for technology stack and fault tolerance, which will facilitate the development of IoT applications on large scale.

While data collected in the IoT generates billion pieces of information with hundreds of data formats, it's hard to classify vast amount of raw data by a complex module in traditional architecture, and it also easily leads to system overload if a large number of redundant data directly goes to the application layer. Scalability and continuous integration. IoT aims to keep on growing all the time and it has generated a lot of complex code. Developers with different technological skills often concern differently and everyone has their own limitations. So a good and independent architecture can offer a mechanism that the programmers can enjoy the freedom of coding with simplified integration In a nutshell, with the development of IoT technology, traditional architecture can't meet the requirements of heterogeneous, interoperable, customizable and scalable systems. To deal with above challenges, it is recommended to decompose the IoT system into Microservices to perform different kinds of tasks.

By using message driven and registry/discovery mechanism in core service, we can build systems that can easily extend, evolve and integrate third party applications to support interoperability and scalability. Moreover, the system uses device plugins to shield the differences of hardware facilities in order to support more heterogeneous platforms.

4.6 Enterprise IoT and Micro Services

Microservices are found suitable in the beginning for large scale Enterprise System development and there are many success stories , Similarly it is perfectly suitable for developing IoT Solutions of Enterprises too. Cloud-hosted Microservices would create an IoT model with less complexities. For example, one set of functions would speak to sensors and controllers and make them visible in the form of data rather than devices. The other set of functions could just process that data and apply some rules to that data. Yet another set of functions could fetch in data from 3rd party enterprise systems like CRM/ERP systems.

Thus, the very nature of IoT is supporting the architectural pattern of Microservices. Microservices also offer a way of scaling the infrastructure both horizontally and vertically giving long term benefits to the IoT deployments. Each of the services can scale based on the needs.

Given the dynamism of deployment and scalability expectations which comes with IoT, Microservices need to become an important part of the overall IoT Strategy.

The following characteristics of Microservices are seamlessly solve the challenges of Enterprise IoT.

Autonomy: Each component service in a microservices architecture can be developed, deployed, operated, and scaled without affecting the functioning of other services. The Enterprise Application Integration challenges were handled with web services which would bridge autonomous systems together which can solve a collective purpose. In IoT from edge to cloud there are several functional and technical requirements which can be composed by autonomous components using Microservices.

Specialized: Since there is no dependency among the services, each service is designed for a set of capabilities and focuses on solving a specific problem alone.

Agilie: Microservices foster an organization of small, independent teams that take ownership of their services. It supports an Agile model of system development where the end product is developed by several iterations done by independent teams which is a fundamental requirement for Enterprise IoT.

Flexible: Enterprise IoT often experiences unpredictability in the demands of the system. Microservices allow each service to be independently scaled to meet demand for the application feature it supports.

Ease of Deployment: Microservices enable continuous integration and continuous delivery, making it easy to try out new ideas and to roll back if something doesn't work. It plays a major role in Devops practise where the development and operations need to be in sync. Quick deployment avoids delay in transforming the technology changes to operations.

Freedom: Teams have the freedom to choose the best tool to solve their specific problems. It is no longer necessary to stick with same set of technologies, platforms or tools throughout the application family for the sake of uniform communication. Microservices enables different business units of an Enterprise can have their own technology choice.

Resilience: Service independence increases an application's resistance to failure. When the success of Enterprise IoT systems are defined based on their up time and revenue generation, Resilience is a non-negotiable feature.

4.7 The trade-off between Monolithic and Micro Services

When it is clearly understood that Companies that start investing in Microservices reap several benefits and their eco system changes very rapidly and move on further towards architectures like messaging, key value stores, and the kinds of things that facilitate communication and data sharing between services, there are down sides too. This may not be suitable to every company and it may be a difficult problem to solve in some environments to build a Microservices ecosystem.

It is easier to construct a monolithic systems since we don't have to worry about the problem of communication among the services and managing them throughout the application life cycle. When two different functionalities of a same system is running in two separate processes, in two separate containers, in separate virtual machines, we must worry about data coordination. So, there are definitely some overheads that we must solve as a result of choosing to go with Microservices.

So the Enterprises should make a decision based on the challenges to be faced while implementing Microservices. Some of the challenges are comparatively trivial or easily handled, whereas others may require new processes or tools for the company. Some of the challenges are given in this section.

Microservices can be less stable than when it gets complex and huge. An individual Microservice can be well-tested before deployment in conjunction with some common configurations and programs, it's practically impossible to test every configuration of Microservices in the real world scenario. As we combine them, they may interact in unforeseen ways.

Security in containers can be an issue. It's not always easy to tell where Microservices reside, which can make securing them a headache. And with different Microservices interacting with each other, this gives hackers more opportunities to penetrate the system. Savvy DevOps teams are moving to a more granular security policy called microsegmentation to get around this problem — but the solution isn't universal yet.

The migration from Monolithic to Microservices is a difficult process and the Enterprises should make a judgement to check if the investment is worth for it. It can be difficult to make the transition. If a company has been producing and maintaining a monolithic application, it will take time and discipline to replace it with a micro services architecture — although many companies have successfully accomplished this, and found the investment worthwhile.

As a summary, the following table provides a comparative study of Monolithic and Micro servies architecture.

	Monolithic	Micro Services
1	Simple to develop, test and deploy in the early stages of any application development. It gets complicated while things becomes huge.	From the early stages, it needs meticulous planning and design. It is not intended for very simple application development tasks.
2	It has limitation in terms of size and complexity.	Scalability is an important aspect in micro services and it adopts well when the application gets complicated.
3	Need to redeploy the entire application on each update.	Due to the independent deployments a huge time is saved without the need to redeploy the entire application.
4	Continuous deployment is difficult.	It is designed for continuous deployment.
5	Reliability is lesser. Bug in a module can bring down the entire application.	The bugs like memory leaks can be effectively isolated and treated.
6	New technologies cannot be adopted	Complete freedom can be

	because of the integration challenges. So there are constraints in terms of choosing the development technology.	experienced in terms of technology selection for a service. As they are deployed separately the services can talk to each other and the development teams have their freedom to choose the most effective technology stack for a particular service.
7	Database operations are comparatively simpler due to the centralized database of a monolithic architecture.	Microservices has the partitioned database architecture. Business transactions that update multiple business entities in a microservices-based application need to update multiple databases owned by different services. Using distributed transactions is usually not an option and you end up having to use an eventual consistency based approach, which is more challenging for developers.

Table 4.1: Comparison of Monolithic and Microservices approaches

This chapter introduces the idea of Enterprise IoT with all its challenges and possibilities. The consumer IoT cannot be expected to reward in proportion with the amount of Research and Development….