IoT Middleware Device & Security Protocols

By

Zafrin Malek Mithila

ID: 2017-1-98-002

A Research Project Submitted in Partial Fulfillment of the Requirements for the Degree of Masters of Science in Telecommunication Engineering

DEPARTMENT OF ELECTRONICS & COMMUNICATIONS ENGINEERING

EAST WEST UNIVERSITY

DECEMBER 2017

APPROVAL

The Research Project titled “IoT Middleware Device and Security Protocols” submitted by Zafrin Malek Mithila (SID: 2017-1-98-002) to the Department of Electronics & Communications Engineering, East West University, Dhaka, Bangladesh has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Masters of Science in Telecommunications Engineering and approved as to its style and contents.

Approved By

(Supervisor)

Dr. Mohammad Arifuzzaman

Assistant Professor

ECE Department

East West University

Dhaka, Bangladesh

DECLARATION

I hereby, declare that the work presented in this Research Project is the outcome of the investigation performed by me under the supervision of Dr. Mohammad Arifuzzaman, Assistant Professor, Department of Electronics & Communications Engineering, East West University, Dhaka, Bangladesh. I also declare that no part of this Research Project and thereof has been or is being submitted elsewhere for the award of any degree or diploma.

Countersigned	**Signature**
--------------------------------	--------------------------------
Dr. Mohammad Arifuzzaman	Zafrin Malek Mithila
(Supervisor)	(SID: 2017-1-98-002)

Abstract

The Internet of Things (IoT) presents a dream of a future Internet where people, computing systems and everyday objects or things collaborate sensing and actuating abilities to integrate a connection between the physical world and the cyber world. A key innovation in the acknowledgment of IoT frameworks is middleware, which is generally depicted as a product framework intended to be the mediator between IoT gadgets and applications. All in all, middleware can ease a development process by incorporating heterogeneous computing and communications devices and supporting interoperability inside the differing applications and administrations. Be that as it may, each one of those advantages can happen to colossal dangers of protection privacy and security issues. There are many existing protocols and mechanisms to secure communications in the IoT. In this paper, we give a brief review of IoT and its middleware device and also the security protocols that is needed to secure the security challenges in middleware.

Index

Page

Chapter: 1 Introduction

1.1 Introduction 6

Chapter: 2 IoT

2.1 What is IoT 8

2.2 IoT Architecture 9

2.3 IoT Characteristics 11

2.3.1 Characteristics of IoT Infrastructure 11

2.3.2 Characteristics of IoT Applications 13

2.4 IoT Applications 15

Chapter: 3 IoT Middleware

3.1 Middleware 16

3.2 Types of Middleware 16

3.3 Middleware Requirements 18

3.3.1 Services a Middleware provides 18

3.3.2 Architectural Requirements 21

3.4 Overview of Existing Work 23

Chapter: 4 IoT Security and Protocols

4.1 Security for the Internet of Things 29

4.2 IoT Protocols 30

4.2.1 Data Protocols 31

4.2.2 Communication Protocols 37

4.3 Security Challenges 41

4.4 Security Requirements 42

Chapter: 5 Conclusion

5.1 Conclusion 43

References 44

Chapter 1

Introduction

1.1 Introduction

Internet of Things (IoT) is a collection of ‘things’ with the advance numerous technologies including sensors, actuators, embedded computing and cloud computing. By enabling easy access of and interaction with a wide variety of physical devices or things such as home appliances, surveillance cameras, monitoring sensors, actuators, displays, vehicles, machines and so on, the IoT will foster the development of applications in many different domains, such as home automation, industrial automation, medical aids such as mobile healthcare, elderly assistance; intelligent energy management and smart grids, traffic management, and many others. A number of operating systems have been developed [2]–[7] to support the development of IoT.

There are four essential components of IoT, they are

- Wireless Sensor Networks (WSNs)
- Radio-Frequency Identification (RFID)
- Machine-to-Machine (M2M) communications
- Supervisory Control and Data Acquisition (SCADA)

A fully functional IoT middleware needs to integrate these technologies to support the envisioned different application domains [8]. To date, the majority of the existing IoT middleware proposals [9]–[17] are WSNs centric. Also, the amount of middleware proposals for RFID, M2M communications, and SCADA is limited unlike WSNs.

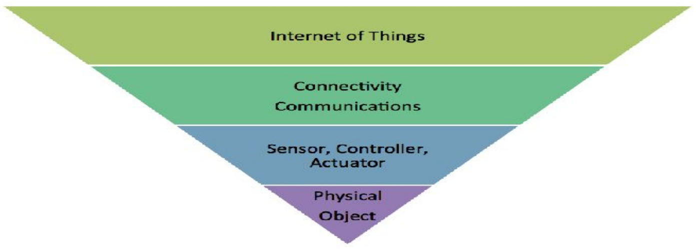

Figure 1: Essential components of IoT

THE INTERNET of Things (IoT) is a domain that represents the next most exciting technological revolution since the Internet [18]–[21]. Smart cities with parking spaces, urban noise, traffic congestion, street lighting, irrigation, and waste can be build with the help of IoT to monitor in real time and manage them more effectively. Energy-efficient smart homes can be build that are more safe and eco-friendly. Devastating disasters such as air and water pollution, earthquakes, forest fires, and many such others calamities can be monitored with smart environments build using IoT. Leaner and smarter manufacturing can be introduced with IoT. We could save numerous human lives with early warnings provided by IoT. Every imaginable industry segment can have massive disruption and innovation. While the IoT offers numerous exciting potentials and opportunities, it remains challenging to effectively manage things to achieve seamless integration of the physical world and the cyber one [18], [22]-[23]. Every day new protocols for connectivity of IoT middleware are being developed and the number is increasing.

This paper aims to provide a clear understanding of current research and challenges of IoT middleware systems and security issues related with that.

The main contributions of this paper are as follows.

1) A classification of the different types of architecture of IoT middleware.

2) A comparative analysis of emerging IoT middleware systems for each architecture type.

3) Iot Protocols detail needed in different layers

4) An assessment of the key research challenges, such as composition, adaptability and security, in building the next generation IoT middleware for rapid composition of IoT applications.

Chapter 2

IoT

2.1 What is IoT

IoT means “A world-wide network of interconnected objects uniquely addressable based on standard communication protocols” [24]. The IoT refers to industrial objects, or “things,” instrumented with sensors, automatically communicating over a network, without human-to-human or human-to-computer interaction, to exchange information and take intelligent decisions with the support of advanced analytics [25].

The definition of “things” in the IoT vision is very wide and includes a variety of physical elements or hardwares. Based on this view of “things,” an enormous number of devices will be connected to the Internet, each providing data and information and some even services. In short, IoT enabled by a myriad of software and hardware technologies.

IoT can be viewed from three perspectives:

1) Internet-oriented;

2) Things-oriented (sensors or smart-things) and

3) Semantic-oriented (knowledge) [21].

Also, the IoT can be viewed as either supporting consumers (human) or industrial applications and indeed could be named as the human Internet of Things (HIoT) or the Industrial Internet of Things (IIoT) [8],[24]-[27].

The first definition of the IoT was from a “things-oriented” perspective, where RFID tags were considered as things [21]. According to the RFID community, IoT can be defined as, “The worldwide network of interconnected objects uniquely addressable based on standard communication protocols” [24].

The European research cluster of IoT (IERC) definition, where “The Internet of Things allows people and things to be connected anytime, anyplace, with anything and anyone, ideally using any path/network, and any service” [28]-[29].

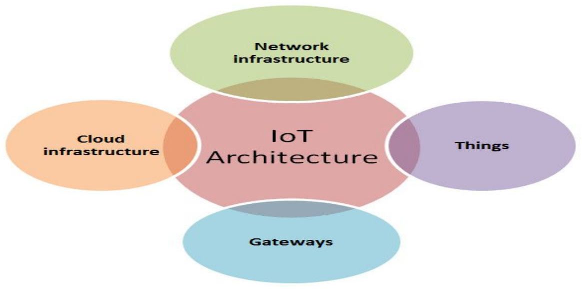

Figure 2: IoT Architecture

The International Telecommunication Union (ITU) views IoT very similarly: “From anytime, anyplace connectivity for anyone, we will now have connectivity for anything” [30].

Finally, we can conclude it as connectivity of physical objects to the Internet for the purposes of monitoring and controlling their behavior to gain efficiencies and create new capabilities.

2.2 IoT Architecture

There are three existing IoT middleware architectures from our observation point. The first type, which refer to it as a service-based solution, generally adopts the service oriented architecture (SOA) [31] and allows developers or users to add or deploy a diverse range of IoT devices as services. The second type, which is known as cloud-based solution, limits the users on the type and the number of IoT devices that they can deploy, but enables users to connect, collect, and interpret the collected data with ease since possible use cases can be determined and programmed a priority. The third type is the actor-based framework that emphasizes on the open, plug and play IoT architecture. Many IoT devices can be made reusable actors and distributed in the network.

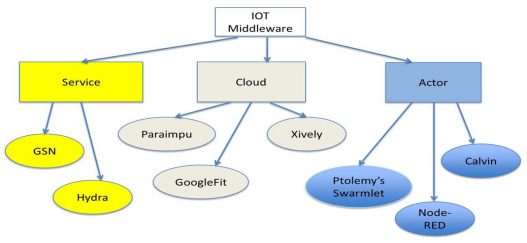

Figure 3: Overview of Various IoT Middleware

We can also represent IoT architecture in four categories of interconnected systems: things, gateways, network & cloud, and services-creation & solutions layers, as shown in Figure 2.

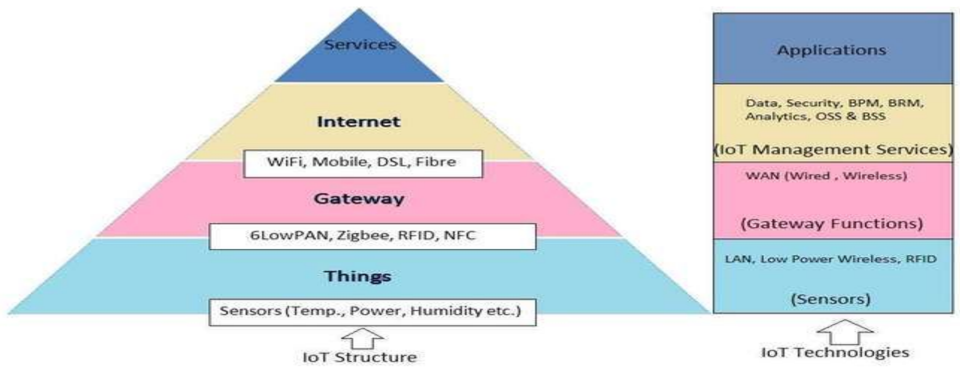
Services
Internet
WiFi, Mobile, DSL, Fibre
Gateway
6LowPAN, Zigbee, RFID, NFC
Things
Sensors (Temp., Power, Humidity etc.)
IoT Structure
Applications
Data, Security, BPM, BRM, Analytics, OSS & BSS
(IoT Management Services)
WAN (Wired , Wireless)
(Gateway Functions)
LAN, Low Power Wireless, RFID
(Sensors)
IoT Technologies

Figure 4: IoT structure and IoT technologies

Things:

Now in our everyday life, billions of things are found in commercial and industrial platforms, in the home, and in the hands of mobile users. Already in cars, device sensors, wearables, mobile phones and not to mention open source development boards equipped with IoT features which can connect directly through broadband or wireless networks and accessing the Internet. A requirement for IoT solutions includes things to either be intelligent so they can filter and manage data locally, or connect to gateways that provide this functionality.

Examples:

- Mobile: smart phones, tablets, GPS systems, wearables
- Home: security alarms, energy consumption monitors, lighting switches, thermostats
- Industrial: smart buildings, factory automation, energy grids, fleets
- Others: Arduino, Raspberry Pi, Intel Edison development boards with equipped sensors

Gateways:

Around almost 85 percent of existing things are not designed to connect to the Internet and so cannot share data with the cloud which is a major barrier to the concept of IoT. To addressing this issue, gateways for mobile, home, and industrial has been introduced to act as intermediaries between legacy things and the cloud, providing the needed connectivity, security, and manageability.

Network and Cloud:

Network Infrastructure:

The Internet is a global system of interconnected IP networks that link computer systems together. This network infrastructure, comprising routers, aggregators, gateways, repeaters and other devices that control data traffic flow also connects to telecom and cable networks (e.g. 3G, 4G/ LTE) operated by service providers.

Data Center / Cloud Infrastructure:

Data centers and cloud infrastructure contain large pools of virtualized servers and storage that are networked together. Supporting IoT, this infrastructure runs applications that analyze data from devices and sensors in order to generate actionable information used for services and decision making.

2.3 IoT Characteristics:

The main characteristics of the IoT are presented from infrastructure and application perspectives.

1. Characteristics of IoT Infrastructure:

Heterogeneous Devices: As Iot d has a nature of embedded and sensor computing services means that low-cost computing platforms are likely to be used. In fact, to minimize the impact of such devices on the environment and energy consumption, low-power radios are likely to be used for connection to the Internet. For example, BLE, 6LowPAN, Zigbee.

Resource-constrained: For embedded computing and sensors a small device form factor is required limiting their processing, memory, and communication capacity. RFID devices or tags may not have any processing capacity or even battery to power them. On the other hand, devices become expensive and larger in form-factor.

Spontaneous interaction: Sudden interactions can take place in IoT applications as objects move around, and come into other objects communication range, leading to the spontaneous generation of events. Typically, in IoT, an interaction with an object means that an event is generated and is pushed to the system without much human attention.

Ultra-large-scale network and large number of events: In an IoT environment, thousands of devices or things may interact with each other even in one local place (e.g., in a building, supermarket and university) which will produce an enormous number of events as normal behavior. In Global perspective, the IoT network will be ultra-large with nodes in the scale of billions and even in trillions. As a result this uncontrolled number of events may cause problems such as event congestion and reduced event processing capability.

Dynamic network and no infrastructure: IoT will integrate devices, many of which will be mobile, wirelessly connected and resource constrained. Mobile nodes within the network able to leave or join anytime they want. Also, nodes can be disconnected due to poor wireless links or battery shortage. These key factors will make the network in IoT highly dynamic. Within such an ad hoc environment, where there is limited or no connection to a fixed infrastructure, it will be difficult to maintain a stable network to support many application scenarios that depend on the IoT. Nodes will need to cooperate to keep the network connected and active.

Context-aware: Context is key in the IoT and its applications. Context-aware computing stores context information related to sensor data, easing its interpretation. A large number of sensors will generate large amounts of data, which will not have any value unless it is analyzed, interpreted, and understood. Context-awareness (especially in temporal and spatial context) plays a vital role in the adaptive and autonomous behavior of the things in the IoT [32]-[33]. Such behavior will help to eliminate human-centric mediation in the IoT, which ultimately makes it easier to perform M2M communication, a core element of the IoT's vision.

Intelligence: According to Intel's IoT vision, intelligent devices or things and intelligent systems of systems are the two key elements of IoT. In IoT's dynamic and open network, these intelligent entities along with other entities such as Web services (WSs), SOA components, and virtual objects will be interoperable and able to act independently based on the context, circumstances, or environments [34]-[35].

Location-aware: Location or spatial information about things (objects) or sensors in IoT is critical, as location plays a vital role in context-aware computing. In a large scale network of things, interactions are highly dependent on their locations, their surroundings, and presence of other entities (e.g., things and people).

Distributed: As like The traditional Internet, IoT is also a globally distributed network. The strong spatial dimension within the IoT makes the network IoT distributed at different scales (i.e., both globally like the Internet, and also locally within an application area).

2. Characteristics of IoT Applications:

Diverse applications: The IoT can offer its services to a large number of applications in numerous domains and environments. These domains and environments can be grouped into (nonexhaustive) domain categories such as: 1) transportation and logistics; 2) healthcare; 3) smart environment (home, office, and plant); 4) industrial; and 5) personal and social domain. Figure, highlights some key application domains for the IoT.

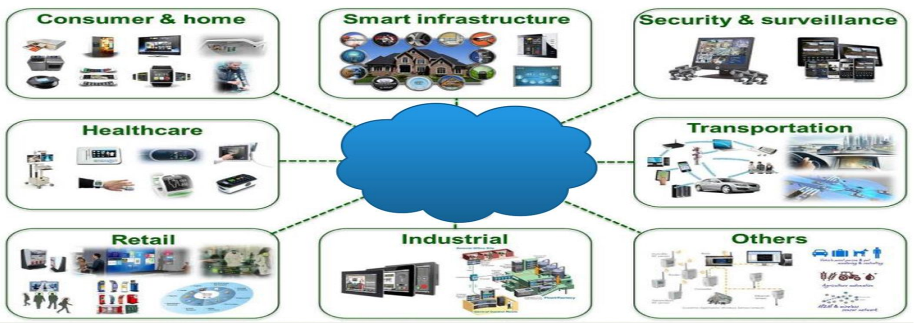

Figure 5: Potential applications of IoT

Real time: Applications using the IoT can be broadly classified as real time and non-real time. For example, IoT for healthcare, transportation will need real-time data or service delivery. Delayed delivery of data can make the application or service useless and even dangerous in mission critical applications.

Everything-as-a-service (XaaS): An everything-as-a-service (XaaS) model is very efficient, scalable, and easy to use [36]. The XaaS model has inspired the sensing as a service approach in WSNs [37]-[38], and this may inevitably lead IoT toward an XaaS model. As more things get connected, the collection of services is also likely to grow, and as they become accessible online, they will be available for use and reuse.

Increased security attack-surface and privacy leakage: Though IoT has huge potential in different domains, applications and networks security is also an issue. Global connectivity and accessibility is required for IoT so that anyone can access it anytime and anyway and also applications may collect information about people's daily activities such as travel routes, buying habits, and daily energy usage etc. This increases the chances of being attacked and exploited in applications and networks. The introduction of cloud computing makes the problem of privacy leakage even worse. The inherent complexity of the IoT further complicates the design and deployment of efficient, interoperable, and scalable security mechanisms.

2.4 IoT Applications:

The fast growth of the number of IoT devices utilized is predicted to reach 41 billion in 2020 with an $8.9 trillion market [39] as stated in the 2013 report of the International Data Corporation. The difference between IoT and the traditional Internet is the absence of Human role. The IoT devices can create information about individual’s behaviors, analyze it, and take action. Services provided by IoT applications offer a great benefit for human’s life, but they can come with a huge price considering the person’s privacy and security protection.

Chapter 3

IoT Middleware

3.1 Middleware

Universal access of computing resource is the core of the Internet of Things, which means incorporating computing and connectivity in all the things around us. Interoperability of such heterogeneous devices needs well-defined standards. But the standardization is difficult because of the varied requirements of different applications and devices. For such heterogeneous applications having a middleware platform is the solution which will abstract the details of the things for applications. It will cover up the details of the smart things and should act as a software bridge between the things and the applications. It needs to provide the required services to the application developers [32] so that they can focus more on the requirements of applications rather than on interacting with the baseline hardware. To summarize, the middleware abstracts the hardware and provides an Application Programming Interface (API) for communication, data management, computation, security, and privacy.

3.2 Types of Middleware:

1. **Message Oriented Middleware:** Message-oriented middleware comprises a category of inter-application communication software that usually relies on asynchronous message-passing, as opposed to a request-response architecture. In case of asynchronous systems, message queues provide temporary storage when the destination program is busy or unable to get connected. It enables applications to be disbursed over various platforms and makes the process of creating software applications spanning many operating systems and network protocols much less complicated. It has many advantages over middleware

alternatives (e.g. hard coding logic) and is one of the most widely used types of middleware.

2. **Object Middleware:** Object middleware, also known as object request broker, gives applications the ability to send objects and request services via an object oriented system. In short, it manages the communication between objects.
3. **Remote Procedure Call (RPC) Middleware:** An RPC calls procedures on remote systems and is used to perform synchronous or asynchronous interactions between applications or systems. It is usually utilized within a software application.
4. **Database Middleware:** This type of middleware allows for direct access to databases providing direct interaction with them. There are a lots of database gateways and connectivity options available. You simply have to see what will work best for your necessary solution. This is the most general and commonly known type of middleware. This includes SQL database software.
5. **Transaction Middleware:** This type of middleware includes applications like transaction processing monitors. It also encompasses web-application servers. These types of middleware are becoming more and more common today.
6. **Portals:** This refers to enterprise portal servers. It is considered middleware because portals facilitate front-end integration. They are used to create interactions between a user’s computer or device and back-end systems and services.
7. **Embedded Middleware:** This type of middleware allows for communication and integration services with an interface of software or firmware. It acts as a liaison between embedded applications and the real-time operating system.
8. **CONTENT-CENTRIC MIDDLEWARE:** This type of middleware allows you to abstract specific content without worrying how it is obtained. This is done through a simple provide/consume abstraction. It is similar to publish/subscribe middleware, which is another type of this software that is often used as a part of web-based applications.

3.3 Middleware Requirements

A middleware provides a layer of software between applications, the operating system and the network communications layers. It facilitates and coordinates some aspect of co-operative processing. So from the computing perspective, a middleware provides a layer between application software and system software. In these requirements are grouped into two sets:

1. The services such a middleware should provide and
2. The system architecture should support.

Functional requirements capture the services or functions (e.g. abstractions, resource management) a middleware provides and nonfunctional requirements (e.g. reliability, security, and availability) capture QoS support or performance issues.

1. **Services a Middleware provides:**

   As the focus is on generic or common functional ones, as follows.

   Resource discovery: IoT resources include heterogeneous hardware devices (e.g. RFID tags, sensors, sensor mote, and smartphones), devices power and memory, analogue to digital converter devices (ADC), the communications module available on those devices, and infrastructural or network level information (e.g. network topology and protocols), and the services provided by these devices. Assumptions related to global and deterministic knowledge of these resources' availability are invalid, as the IoT infrastructure and environment is dynamic. This is a different model to centralized distributed systems, where resource publication, discovery, and communication are generally managed by a dedicated server. Discovery mechanisms also need to scale well, and there should be efficient distribution of discovery load, given the IoT composition of resource-constrained devices.

   Resource management: An acceptable QoS is expected for all applications, and in an environment where resources that impact on QoS are constrained, such as the IoT, it is important that applications are provided with a service that manages those resources. This means that resource usage should be monitored, resources allocated or provisioned in a fair manner and resource conflicts resolved. In IoT architectures, especially in service oriented or virtual machine (VM) based architectures, middleware needs to facilitate potentially spontaneous resource (service) (re)composition, to satisfy application needs.

Data management: Data are key in IoT applications. In the IoT, data refer mainly to sensed data or any network infrastructure information of interest to applications. An IoT middleware needs to provide data management services to applications, including data acquisition, data processing (including preprocessing) and data storage. Preprocessing may include data filtering, data compression and data aggregation.

Event management: There are potentially a massive number of events generated in IoT applications, which should be managed as an integral part of an IoT middleware. It should provide real-time analysis of high-velocity data so that downstream applications are driven by accurate, real-time information and intelligence.

Code management: Deploying code in an IoT environment is challenging and should be directly supported by the middleware. In particular, code allocation and code migration services are required. Code allocation selects the set of devices or sensor nodes to be used to accomplish a user or application level task. Code migration transfers one node/device's code to another one potentially reprogramming nodes in the network.

Key nonfunctional requirements of IoT middleware are as follows.

Scalability: An IoT middleware needs to be scalable to accommodate growth in the IoT network and applications or services. Considering the size of the IoT network IPv6 is a very scalable solution for addressability, as it can deal with a huge number of things that need to be included in the IoT [21]. Loose coupling and/or virtualization in middleware is useful in improving scalability, especially application and service level scalability, by hiding the complexity of the underlying hardware or service logic and implementation.

Real time or timeliness: Real time data providing is must service a middleware should provide as the correctness of an operation that supports depends not only on its logical correctness but also on the time in which it is performed. As the IoT will deal with many real-time applications (e.g. transportation, healthcare), on-time delivery of information or services in those applications is critical. Delayed information or services in such applications can make the system useless and even dangerous.

Reliability and Availability: The reliability and availability requirements should work together to ensure the highest fault tolerance required from an application. The middleware reliability ultimately helps in achieving system level reliability and also in critical ones, must be available or appear available, at all times. Every component or service in a middleware needs to be reliable to achieve overall reliability, which includes communication, data, technologies and devices from all layers even in the presence of failures. Even also if there is a failure somewhere in the system, its recovery time and failure frequency must be small enough to achieve the desired availability.

Security and privacy: Security is critical to the operation of IoT. In IoT middleware, security needs to be considered in all the functional and nonfunctional blocks including the user level application. Context-awareness in middleware may disclose personal information (e.g. the location of an object or a person). Like security, every block of middleware, which uses personal information, needs to preserve the owner’s privacy.

Ease-of-deployment: Since an IoT middleware is typically deployed by the user, deployment should not require expert knowledge or support. Complicated installation and setup procedures must be avoided.

Popularity: An IoT middleware should be continuously supported and extended which is not necessarily a requirement but as a large number of users who adopt a particular technology motivate future testing and development. Usually, this facility is provided within a community of developers and researchers.

2. Architectural Requirements:

The architectural requirements are designed to support application developers. They include requirements for programming abstractions and other implementation-level concerns.

Programming abstraction: Providing an API for application developers is an important functional requirement for any middleware. For the application or service developer, high-level programming interfaces need to isolate the development of the applications or services from the operations provided by the underlying, heterogeneous IoT infrastructures. The level of abstraction, the programming paradigm and the interface type all need to be considered when defining an API. The level of abstraction refers to how the application developer views the system (e.g. individual node/device level, system level). The programming paradigm (e.g. publish/subscribe) deals with the model for developing or programming the applications or services. The interface type defines the style of the programming interface. For instance, descriptive interfaces offer SQL-like languages for data query [40], XML-based specification files for context configuration [41].

Interoperable: A middleware should work with heterogeneous devices/technologies/applications, without additional effort from the application or service developer. Heterogeneous components must be able to exchange data and services. A network should exchange information across different networks, potentially using different communication technologies. Syntactic interoperation should allow for heterogeneous formatting and encoding structures of any exchanged information or service. Semantic interoperability refers to the meaning of information or a service and should allow for interchange between the ever growing and changing set of devices and services in IoT. Meaningful information about services will be useful for the users in composing multiple services as semantic data can be better understood by “things” and humans compared to traditional protocol descriptions [42]-[43].

Service-based: A middleware architecture should be service-based to offer high flexibility when new and advanced functions need to be added to an IoT middleware. A service-based middleware provides abstractions for the complex underlying hardware through a set of services (e.g. data management, reliability, security) needed by applications. All these and other advanced services can be designed, implemented, and integrated in a service-based framework to deliver a flexible and easy environment for application development.

Adaptive: A middleware needs to be adaptive so that it can evolve to fit itself into changes in its environment or circumstances. In the IoT, the network and its environment are likely to change frequently. In addition, application-level demands or context are also likely to change frequently. To ensure user satisfaction and effectiveness of the IoT, a middleware needs to dynamically adapt or adjust itself to fit all such variations.

Context-aware: Context-awareness is a key requirement in building adaptive systems and also in establishing value from sensed data. The IoT middleware architecture needs to be aware of the context of users, devices and the environment and use these for effective and essential services offerings to users.

Autonomous: It means self-governed. Devices/technologies/applications are active participants in the IoT processes and they should be enabled to interact and communicate among themselves without direct human intervention [44]-[45]. Use of intelligence including autonomous agents, embedded intelligence [46], predictive, and proactive approaches (e.g. a prediction engine) in middleware can fulfil this requirement [47].

Distributed: A large-scale IoT systems applications/devices/users (e.g. WSNs and vehicular ad hoc networks) exchange information and collaborate with each other. Such applications/devices/users are likely to be geographically distributed, and so a centralized view or middleware implementation will not be sufficient to support many distributed services or applications. A middleware implementation needs to support functions that are distributed across the physical infrastructure of the IoT.

3.4 OVERVIEW OF EXISTING WORK

Middleware in IoT is a very active research area. Many solutions have been proposed and implemented which are highly diverse in their design approaches (e.g. event-based, database), level of programming abstractions (e.g. local or node level, global or network level) and implementation domains (e.g. WSNs, RFID, M2M, and SCADA).

The existing middleware solutions are grouped for discussion based on their design approaches, as follows:

1. Event-based
2. Service-oriented
3. VM-based
4. Agent-based
5. Tuple-spaces
6. Database-oriented
7. Application-specific

Some middleware use a combination of different design approaches. For instance, many service-oriented middlewares (SOMs) (e.g. SOCRADES and Servilla) also employ VMs in their design and development. Typically, hybrid approaches perform better than their individual design categories by taking the advantages of multiple approaches.

1. **Event-Based Middlewares:** In event-based middleware, components, applications, and all the other participants interact through events. Each event has a type, as well as a set of typed parameters whose specific values describe the specific change to the producer’s state. Events are propagated from the sending application components (producers), to the receiving application

components (consumers). An event system (event service) may consist of a potentially large number of application components (entities) that produce and consume events [48]. Message-oriented middleware (MOM) is a type of event-based middleware. Typically, event-based middleware uses the publish/subscribe pattern. This model contains a set of subscribers and a set of publishers. Subscribers are provided with access to publishers data streams through a common database and they are registered for particular events. The notifications about the events are subsequently and asynchronously sent to the subscribers [49], [10]. This design approach addresses nonfunctional requirements such as reliability, availability, real-time performance, scalability, and security [50]. Event-based middlewares are appropriate in systems where mobility and failures are common. A main advantage of this approach is support for strong decoupling of producers and subscribers. Although many challenges are addressed by most of the event-based middlewares, their support is not totally satisfactory, in particular, interoperability, adaptability, timeliness, and context-awareness are not adequately addressed.

2. **Service-Oriented Middlewares:** The service-oriented design paradigm builds software or applications in the form of services. Service-oriented computing (SOC) is based on service-oriented architecture (SOA) approaches and has been traditionally used in corporate IT systems. The characteristics of SOC, such as technology neutrality, loose coupling, service reusability, service composability, and service discoverability [51], are also potentially beneficial to IoT applications. However, IoT's ultra-large-scale network, resource-constrained devices and mobility characteristics make service discovery and composition challenging. An SOM has the potential to alleviate these challenges through the provision of appropriate functionalities for deploying, publishing/discovering, and accessing services at runtime. SOM also provides support for adaptive service compositions when services are unavailable. A large number of service-oriented IoT middlewares are available. These middlewares can be categorized as

standalone SOM for IoT [52]-[56] or middleware services provided by cloud computing platform as a service (PaaS) model [57]-[58]. Most existing SOMs are WSNs-centric and their scale is limited to WSNs, which is typically in the range of thousands, much less than the ultra large scale (billions) of IoT. A global, scalable, understanding of IoT services syntax, and semantics is required. Most existing standalone SOMs offer only limited security through authentication. Also, cloud platform storage security and trust could be a concern for many IoT applications.

3. **VM-Based Middlewares:** VM-oriented middleware design provides programming support for a safe execution environment for user applications by virtualizing the infrastructure. The applications are divided into small separate modules, which are injected and distributed throughout the network. Each node in the network holds a VM, which interprets the modules. This approach addresses architectural requirements such as high level programming abstractions, self-management, and adaptivity, while supporting transparency in distributed heterogeneous IoT infrastructures [59]-[60]. VMs can be divided into two categories: 1) middleware-level VMs (VMs are placed between the OS and applications) and 2) system-level VMs (substitute or replace the entire OS) [10], [59]. Middleware level VMs add capabilities (e.g. concurrency) to the underlying OSs [61]. System level VMs free up resources that would otherwise be consumed by the OS. The resource-constrained characteristics of WSNs raise an important limitation; VMs require significant memory and processing power resources, which makes virtualization feasible only on resource-rich devices [10]. Moreover, the new languages and tools that need to be adopted create a steep learning curve for users and developers [62].

   Application-specific virtual machines (ASVMs) [61] solve the problems imposed by traditional VM solutions by limiting the generality of the VMs to subsets relevant to application domain(s) [63]. This type of VM minimizes overhead by reducing the size of the interpreted code and by using an on-the-fly compiler to native code. On the hardware side, the interpretation overhead is minimized using CPU-specific bytecode.

4. **Agent-Based Middlewares:** In the agent-based approach to middleware, applications are divided into modular programs to facilitate injection and distribution through the network using mobile agents. While migrating from one node to another, agents maintain their execution state. This facilitates the design of decentralized systems capable of tolerating partial failures [64]. Previous research in this area has presented a number of advantages for using mobile agents in generic distributed systems [65]-[66]. In the context of the IoT middleware requirements, these are: resource management (network load reduction and network latency reduction), code management (asynchronous and autonomous execution and protocol encapsulation), availability and reliability (robustness and fault-tolerance), adaptiveness and heterogeneity [67]. Moreover, an agent can engage in dialogues with other software agents to proactively gather data and update only parts of the application. Additionally, agent-based approaches consider resource-constrained devices [68].

   The IoT vision is to support the connection of various physical world objects to a common infrastructure and designing a system that will enable this is a complex process. The use of agent-based systems can reduce the complexity of designing such systems by defining some higher-level policies rather than direct administration. However, the autonomous characteristic of agents can lead to unpredictability in the system at runtime. The patterns and the effects of their interactions are uncertain [69]. Moreover, mobile agents are susceptible to message loss, especially in resource-constrained environments [70]. This imposes many limitations for an IoT middleware solution, including the ability to perform code management tasks.

5. **Tuple-Space Middlewares:** In tuple-space middlewares, each member of the infrastructure holds a local tuple space structure. A tuple space is a data repository [71] that can be accessed concurrently. All the tuple spaces form a

federated tuple space on a gateway. This approach suits mobile devices in an IoT infrastructure, as they can transiently share data within gateway connectivity constraints. Applications communicate by writing tuples in a federated tuple space and by reading them through specifying the pattern of the data they are interested in.

6. **Database-Oriented Middlewares:** A sensor network is viewed as a virtual relational database system in database-oriented middleware. Applications can make query to the database using an SQL-like query language, which enables the formulation of complex queries [10].

   A database approach to middleware views the whole network as a virtual database system. Easy-to-use interfaces support user queries to sensor networks to extract data of interest. However, only approximate results are returned. Most IoT applications are real time, where time and space are important. Database middlewares do not support timeliness. Energy consumption is reduced by collecting data from individual nodes. While database middlewares can provide good programming abstraction support and have good data management support, the rest of IoT middleware requirements are mostly ignored. Moreover, database middleware approach uses a centralized model, which makes it difficult to handle large-scale sensor networks dynamics. Furthermore, database middlewares generally do not support hard real-time applications, and are not popular with technology adopters, whose main interest tends to be in safety critical systems.

7. **Application-Specific Middlewares:** An application-specific (i.e., application-driven) approach to middleware focuses on resource management support (i.e., QoS support) for a specific application or application domain by implementing an architecture that fine-tunes the network or infrastructure (as shown in Fig. 11) based on the application or application domain requirements.

   Application-specific solutions do not address the heterogeneity of an IoT infrastructure as there is tight coupling between applications and middleware layer. Moreover, the application specific approach creates only specialized middleware solutions [72] instead of general purpose solutions. This does not satisfy the IoT middleware requirements since an IoT solution should support multiple applications. Furthermore, all the presented application-specific middleware solutions use a centralized resource discovery mechanism, which is not a viable approach for a distributed fault-tolerant IoT solution. Moreover, these drawbacks make this type of middleware solutions unattractive to technology adopters.

Chapter 4

IoT Security and Protocols

4.1 IoT Security Challenges

Services given by IoT applications offer an incredible advantage for human's life but they can come with a huge cost considering the person's privacy and security assurance. Sicari et al. [73] presented research challenges and the current solutions in the field of IoT security focusing on the main security issues which were identified in eight categories:

1. Authentication
2. Access control
3. Confidentiality
4. Privacy
5. Trust
6. Secure middleware
7. Mobile security
8. Policy enforcement

As the communications infrastructures of internet evolves to encompass sensing objects, appropriate mechanisms will be necessary to secure communications with such devices, in the context of future IoT applications, in areas as diverse as healthcare (e.g. remote patient monitoring or monitoring of elderly people), smart grid, home automation (e.g. security, heating and lightening control) and smart cities (e.g. distributed pollution monitoring, smart lightenning systems) among many others. Such technologies currently form a much necessary wireless communications protocol stack for the IoT that together with the various communication technologies, is analyzed in detail in [74] the paper. This stack is enabled by the technologies the industry believes to meet the important criteria of reliability, power-efficiency and Internet connectivity.

As standardized communication protocol is our focus for the IoT, our discussion is guided by the protocol stack enabled by the various IoT communication protocols available or currently being designed.

The current Internet architecture, IP-based communication protocols will play a key role in enabling the ubiquitous connectivity of devices in the context of IoT applications. Such communication technologies are being developed in line with the constraints of the sensing platforms likely to be employed by IoT applications, forming a communications stack able to provide the required power—efficiency, reliability and Internet connectivity. As security will be a fundamental enabling factor of most IoT applications, mechanisms must also be designed to protect communications enabled by such technologies.

Security Requirements

Security of IoT communications may be addressed in the context of the communication protocol itself or on the other end by external mechanisms.

Other security requirements must also be considered for the IoT and in particular regarding communications with sensing devices. For example, WSN environments may be exposed to Internet-originated attacks such as Denial of Service (DoS) and in this context availability and resilience are important requirements. Mechanisms will also be required to implement protection against threats to the normal functioning of IoT communication protocols. Other relevant security requirements are privacy, anonymity, liability and trust which will be fundamental for the social acceptance of most of the future IoT applications employing Internet integrated sensing devices.

4.2 IOT PROTOCOLS

The set of IoT protocols, extends IoTs from a single constrained device to a whole range of cloud systems, that allow devices and servers to talk to one another. For this purpose, IEEE (Institute of Electrical and Electronics Engineers) and ETSI (European

Telecommunications Standards Institute) have defined some of the most important IoT protocols.

Because of the constraints of sensing platforms and the scale factors of the IoT typically make most of the communications and security solutions implemented in the Internet is not suited for the IoT, working groups formed at standardization bodies as the Institute of Electrical and Electronics Engineers (IEEE) and the Internet Engineering Task Force (IETF) are designing new communications and security protocols that will play a fundamental role in enabling future IoT applications. Such technological solutions are being designed in line with the constraints and characteristics of low-energy sensing devices and low rate wireless communications.

QoS control is a much better metric than the overloaded “real-time” term. QoS control refers to the flexibility of data delivery. A system with complex QoS control may be harder to understand and program, but much more demanding applications can be build.

For example, consider the reliability QoS. Most protocols run on top of TCP which delivers strict, simple reliability. Every byte put into the pipe must be delivered to the other end, even if it takes many retries. This is simple and handles many common cases, but it doesn’t allow timing control. TCP’s single-lane traffic backs up if there’s a slow consumer.

4.2.1 Data Protocols

The IoT needs many protocols as in Iot devices must communicate with each other (D2D) with a motive to collect and send data to the server infrastructure (D2S) which will ultimately have an infrastructure to share the data (S2S) providing it back to the devices to analysis programs.

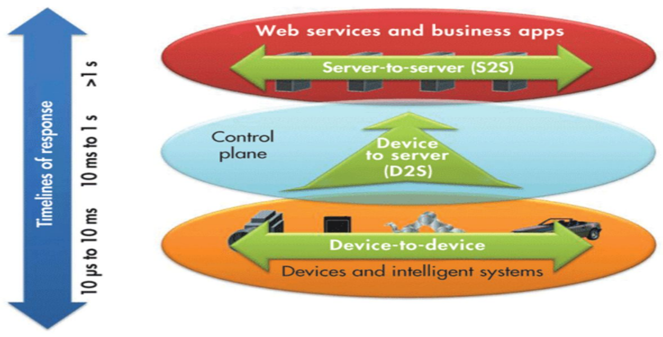

Figure 6: IoT protocols need to address response time

The Fig. 6 shows that within 10 microsecond to 10 millisecond the communication between devices take place and sends the data or information to the server within less than 1 second. The task is not over here, then servers communicate between themselves to provide better services by analysis the data and give a proper way out. D2D communication requires real time communication guaranteed where S2S communication do not required super time communication. This is the reason for being S2S communication is greater than 1 second.

DDS:

The Data Distributed Service is a fast bus for integrating intelligent machines (D2D). It distributes data to other devices that directly use device data. In short it can efficiently deliver millions of messages per second to many simultaneous receivers.

- Devices are fast even “Real time” is often measured in microseconds.
- DDS implements direct device-to-device “bus” communication with a relational data model. So, Hub-and-spoke is completely inappropriate for device data use.
- Devices need to communicate with many other devices in complex ways so TCP’s simple and reliable point-to-point streams are not suitable.
- Instead, DDS offers detailed quality-of-service (QoS) control, multicast, configurable reliability, and pervasive redundancy.
- Besides, fan-out is a key strength. DDS offers powerful ways to filter and select exactly which data goes where and “where” can be thousands of simultaneous destinations. Some devices are small so there are lightweight versions of DDS that run in constrained environments.

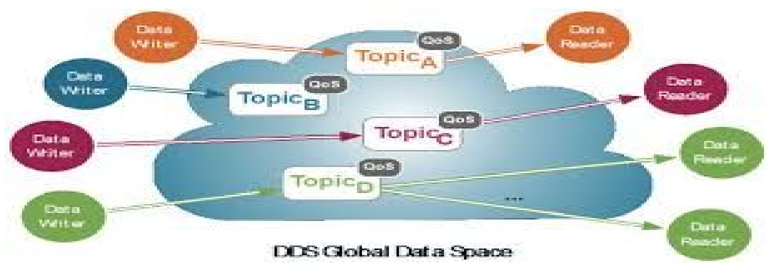

Figure 7: DDS global data space

CoAP:

Constrained Application Protocol (CoAP) was created by the IETF Constrained RESTful Environments (CoRE) working group. CoAP is an Internet application protocol for constrained devices that can be thought of as an alternative to HTTP. It is designed to be used between devices on the same constrained network between devices and general nodes on the Internet and between devices on different constrained networks both joined on the Internet. This protocol is especially designed for IoT systems based on HTTP protocols.

- CoAP unlike HTTP, it incorporates optimizations for constrained application environments [40].
- It uses the EXI (Efficient XML Interchanges) data format, which is a binary data format and is far more efficient in terms of space as compared to plain text HTML/XML.
- Other supported features are built in header compression, resource discovery, autoconfiguration, asynchronous message exchange, congestion control and support for multicast messages.
- There are four types of messages in CoAP: non-confirmable, confirmable, reset (nack) and acknowledgement.
- For reliable transmission over UDP, confirmable messages are used [48].

- CoAP is helping to minimize the cost of cloud–device connections, enabling IoT devices to cost effectively and securely send data over large distances while consuming very little power.

Low Power Wide-Area-Networks (LPWAN):

LPWAN is a protocol for long range communication in power constrained devices. The LPWAN class of protocols is low bit-rate communication technologies for following IoT scenarios [78].

- Narrow band IoT: It is a technology made for a large number of devices that are energy constrained and necessary to reduce the bit rate. This protocol can be deployed with both the cellular phone GSM and LTE spectra.
- Sigfox: It is one more protocol that uses narrow band communication (10 MHz) focuses on using very long waves. It uses free sections of the radio spectrum (ISM band) to transmit its data. Thus, the range can increase to a 1000 kms.
- Weightless: It uses a differential binary phase shift keying based method to transmit narrow band signals. To avoid interference, the protocol hops across frequency bands (instead of using CSMA). It supports cryptographic encryption and mobility. Along with frequency hopping, two additional mechanisms are used to reduce collisions. The downlink service uses time division multiple access (TDMA) and the uplink service uses multiple subchannels that are first allocated to transmitting nodes by contacting a central server..
- Neul: This protocol operates in the sub-1 GHz band. It uses small chunks of the TV white space spectrum to create low cost and low power networks with very high scalability. It has a 10 km range and uses the Weightless protocol for communication.
- LoRaWAN: This protocol is similar to Sigfox. It targets wide area network applications and is designed to be a low power protocol. Its data rates can vary from 0.3 kbps to 50 kbps and it can be used within an urban or a suburban

environment (2–5 kms range in a crowded urban area). It was designed to serve as a standard for long range IoT protocols.

MQTT:

MQTT, the Message Queue Telemetry Transport is a protocol for collecting device data and communicating it to servers (D2S). It targets device to collect datas through a distributed network. It has a hub-and-spoke architecture.

- As it is a D2S communication protocol so all the devices are connected to a data concentrator server.
- In IoT data collection is huge and misuse of data is not desirable, so the protocol works on the top of TCP which provides a simple and reliable stream.
- Since the IT infrastructure uses the data, the entire system is designed to easily transport data into enterprise technologies like ActiveMQ and enterprise service buses (ESBs).

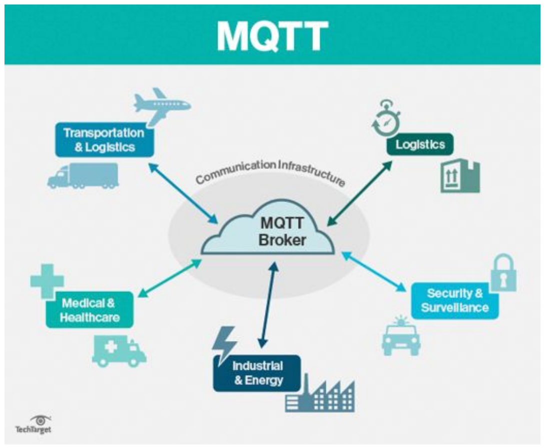

Figure 8: Communication infrastructure with MQTT

XMPP:

XMPP, Extensible Messaging and Presence Protocol is a protocol best for connecting devices to people in short it provides a great way to connect devices to server. It was originally called "Jabber" and was developed for instant messaging (IM) to connect people to other people via text messages. Its strengths in addressing, security and scalability make it ideal for consumer-oriented IoT applications.

- For making person-to-person communications natural, XMPP uses the XML text format. It also runs over TCP or perhaps over HTTP on top of TCP.
- In the IoT context, XMPP offers an easy way to address a device. This is especially handy if that data is going between distant, mostly unrelated points, just like the person-to-person case.
- Its key strength is a 'name@domain.com' addressing scheme that helps connect the needles in the huge Internet haystack.
- It is not designed to be fast, "real time" to XMPP is on human scales, measured in seconds. In fact, most implementations use polling or checking for updates only on demand. A protocol called BOSH (Bidirectional streams over Synchronous HTTP) lets servers push messages.

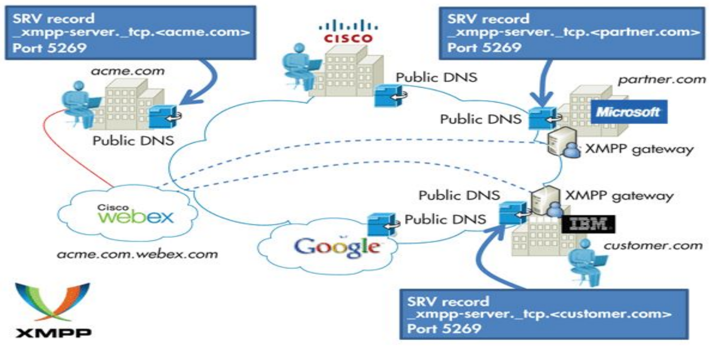

Figure 9: Inter-Domain Federation

AMQP:

The Advanced Message Queuing Protocol (AMQP) is all about queues where a queuing system designed to connect servers to each other (S2S). It sends transactional messages between servers and is most appropriate for the control plane or server-based analysis functions.

- As a message-centric middleware that arose from the banking industry, it can process thousands of reliable queued transactions as it is focused on not losing messages.

- Communications from the publishers to exchanges and from queues to subscribers use TCP, which provides strictly reliable point-to-point connection.
- True to its origins in the banking industry, AMQP middleware focuses on tracking all messages and ensuring each is delivered as intended, regardless of failures or reboots.
- For ensuring delivery, endpoints acknowledge acceptance of each message.
- The standard also describes an optional transaction mode with a formal multiphase commit sequence.

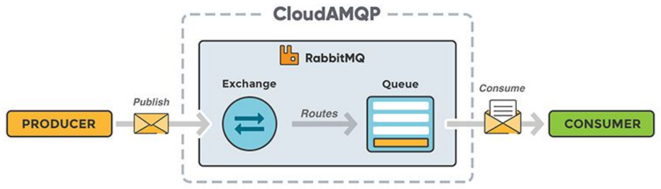

Figure 10: RabbitMQ message broker process.

4.2.2 Communication Protocols

BLE

Bluetooth Low Energy IEEE 802.11 channel is designed for lower data throughput and reduced the power consumption using 2.4 GHz ISM band, 40 Channels on 2 MHz spacing.

- BLE protocol stack is composed of two main parts: the Controller and the Host.
- The Controller comprises the Physical Layer and the Link Layer and is typically implemented as a small System-on-Chip (SOC) with an integrated radio.
- Communication between the Host and the Controller is standardized as the Host Controller Interface (HCI).
- A Bluetooth Low Energy device can communicate with the outside world in two ways broadcasting or connections.
- Everything is Optimized for Lowest Energy, by reducing transmit peak current and received time by sending short packets.
- It is designed for sending small chunks of data (exposing state).

- Data can triggered by local events and can be read at anytime by a client.
- Can theoretically support an unlimited number of devices, but the practical number of simultaneously connected devices is between 10 and 20.
- For security it uses the Cipher Block Chaining-Message Authentication Code (CCM) algorithm for authentication and also uses a 128-bit AES block cipher for encryption

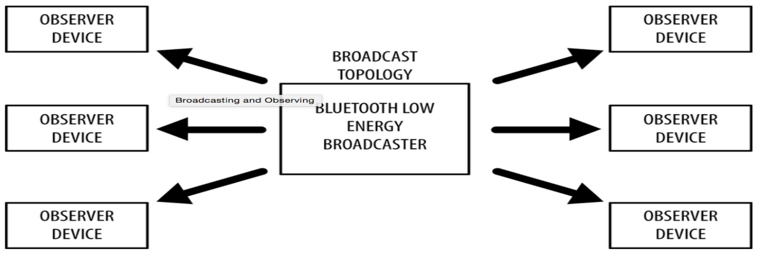

Figure 11: Broadcast topology

Bluetooth

Bluetooth is a Short Range Wireless System, typically a few meters but can go up to approximately 100 meters in a open environment. Dynamically created Bluetooth devices to find and set-up temporary (ad-hoc) connections (small networks) with other devices.

- Bluetooth is a wireless personal area network (WPAN).
- Each Bluetooth wireless network can contain up to 7 active devices and is called a Piconet which can be linked to each other (overlap) to form larger area Scatternets.
- The system control for Bluetooth requires one device to operate as the coordinating device called master while all of the other devices are slaves.
- The characteristics of Bluetooth include an unlicensed frequency band that ranges from 2.4 GHz to 2.483 GHz.

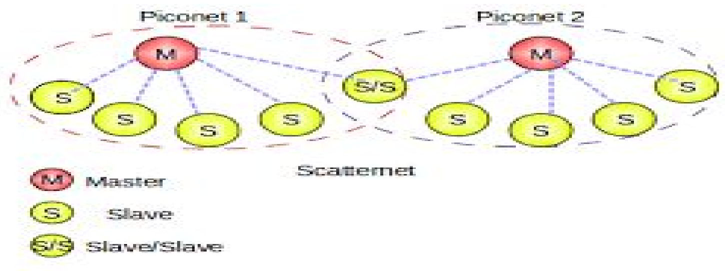

Figure 12: Illustration of Bluetooth Piconet and Scatternet with Slave/Slave node

- Bluetooth have Sleep mode that is a process used in a radio receiver where electronic circuits (such as a receiver) are temporarily deactivated or put into a low power consumption mode (such as backlighting off) to save battery energy.

Zigbee

Zigbee is a standardized wireless protocol for personal area networking which is a hardware and software standard built IEEE 802.15.4 standard that defines the physical (PHY) and Medium Access Control (MAC) Layers. The ZigBee Alliance has added Network (NWK) and application (APL) layer specifications to complete what is called the ZigBee stack. It globally assigned 2.4 GHz unlicensed band or one of the 900 MHz regional bands.

- It provides 10 to 75 meter range. Typical 2.4 GHz hardware has demonstrated 30 meters indoors and over 100 meters outdoors.
- 10 to 115 kbps data throughput.
- Up to 100 ZigBee networks should be able to be co-located and still function.
- ZigBee devices will typically operate at 0.1 to 1% duty cycles.
- This allows the Carrier Sense Multiple Access (CSMA) scheme to produce robust results. ZigBee or 802.15.4 devices listen for a clear channel before they transmit.
- The CSMA algorithm is part of the 802.15.4 software so the user is relieved of the burden of creating a collision avoidance scheme.
- 2 years of battery life from standard AA and AAA batteries.
- Three devices specified

1. Reduced Function Device (RFD)
2. Full Function Device (FFD)
3. Network coordinator (FFDC)

- Mesh topology gives routers and coordinator multiple communication path options.
- ZigBee maintains the low cost while adding the power of mesh networking, a feature not found in most wireless networking standards.

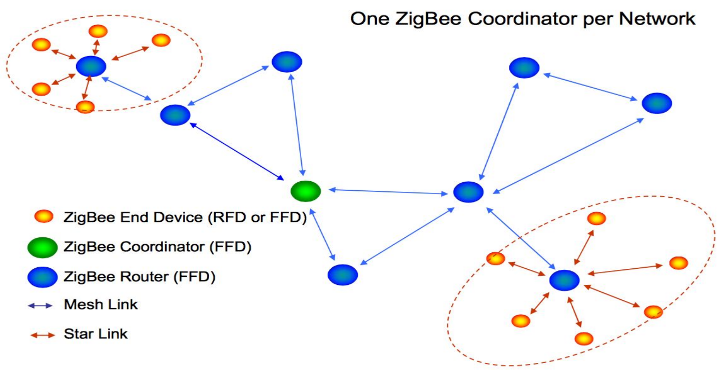
One ZigBee Coordinator per Network
ZigBee End Device (RFD or FFD)
ZigBee Coordinator (FFD)
ZigBee Router (FFD)
Mesh Link
Star Link

Figure 13: ZigBee Network model

Low Power WiFi

The early standards for wireless LAN for WiFi were designed primarily to connect a laptop PC in home, office and on the road. The IEEE 802.11ac is created for connecting IoT devices that would require higher throughputs. The WiFi alliance has recently developed “WiFi HaLow,” which is based on the IEEE 802.11ah standard. It consumes lower power than a traditional WiFi device and also has a longer range. This is why this protocol is suitable for Internet of Things applications. The range of WiFi HaLow is nearly twice that of traditional WiFi.

- Like other WiFi devices, devices supporting WiFi HaLow also support IP connectivity, which is important for IoT applications.

- Let us look at the specifications of the IEEE 802.11ah standard [75] [76]. This standard was developed to deal with wireless sensor network scenarios, where devices are energy constrained and require relatively long range communication.
- IEEE 802.11ah operates in the sub-gigahertz band (900 MHz). Because of the relatively lower frequency, the range is longer since higher frequency waves suffer from higher attenuation. We can extend the range (currently 1 km) by lowering the frequency further; however, the data rate will also be lower and thus the tradeoff is not justified.
- IEEE 802.11ah is also designed to support large star shaped networks, where a lot of stations are connected to a single access point.

Figure 14: WiFi300 Wi-Fi Module

4.3 Security Challenges

Diverse arrangement of real difficulties emerge contingent upon various cases. The following lists a few security challenges and considerations in planning and building IoT devices or systems are given below [77]:

- Typically small, inexpensive devices with little to no physical security.
- Computing platforms may not support complex and evolving security algorithms due to limited security compute capabilities, lack of higher processing power and low CPU cycles.
- Designed to operate autonomously in the field with no backup connectivity if primary connection is lost.

- Mostly installed prior to network availability which increases the overall on-boarding time.
- Requires secure remote management during and after onboarding.
- Scalability and management of billions of entities in the IoT ecosystem.
- Identification of endpoints in a scalable manner.
- Crypto Resilience
    - Embedded devices may outlive algorithm lifetime. For example, Smart meters could last beyond 40 years
    - Crypto algorithms have a limited lifetime before they are broken
- Physical Protection need to be provided as
    - Mobile devices can be stolen
    - Fixed devices can be moved
- Tamper Detection techniques and design
    - Always On: High Poll rate, more energy, quick detection
    - Periodic Poll: Less energy, slower detection
    - On-event Push: Minimal energy, no detection

4.4 Security Requirements

There are seemingly competing, complex security requirements to be deployed on a platform with potentially limited resources [77]:

- Authenticate to multiple networks securely.
- Ensure that data is available to multiple collectors.
- Manage the contention between that data access.
- Manage privacy concerns between multiple consumers.
- Provide strong authentication and data protection (integrity and confidentiality) that are not easily compromised.
- Maintain availability of the data or the service.
- Allow for evolution in the face of unknown risks.

Chapter 5

Conclusion

5.1 Conclusion

The IoT presents numerous benefits to consumers and has the potential to change the ways that people interact with technology. After a brief explanation of IoT and middleware this survey proposes to clarify the difficulties in adopting middleware for IoT development. The paper highlights the concept of IoT, it's requirements and characteristics. It also presentes what middleware is and about it's requirements. An detailed discussion on the IoT protocols is present on the paper. From all exposed difficulties and problems in this research, we realize that the security challenges in IoT platforms, presented at the end of chapter 4, is the difficulty that requires the greatest attention from the IoT developers. Hence we have the enlistment of the security requirements for IoT. Software for IoT involves distribution and data sharing, thus increasing the risks of data theft. From a security and privacy perspective, the introduction of sensors and devices into currently intimate spaces such as the home, the car, wearable objects, or everyday things to detect and share observations about us increasingly deserves special attention and concern. There is no denying the utility of middleware assists IoT development, but we have to be aware of some concerns about the difficulties and problems that this paper covers in its study.

For future researches, there is good opportunity to apply solutions to all problems listed above or to choose security as the problem most relevant to the use of middleware for IoT.

References

1. Mohammad Abdur Razzaque, Marija Milojevic-Jevric, Andrei Palade, and Siobhán Clarke, “Middleware for Internet of Things: A Survey” IEEE INTERNET OF THINGS JOURNAL, VOL. 3, NO. 1, page: 70-95, FEBRUARY 2016
2. Contiki [Online]. Available: http://www.contiki-os.org/
3. Brillo [Online]. Available: https://developers.google.com/brillo/
4. mbed [Online]. Available: http://mbed.org/technology/os/
5. RIOT [Online]. Available: http://www.riot-os.org
6. Embedded Linux [Online]. Available: http://elinux.org
7. OpenWSN [Online]. Available: http://openwsn.atlassian.net
8. H. Zhou, The Internet of Things in the Cloud: A Middleware Perspective,1st ed. Boca Raton, FL, USA: CRC, 2012.
9. K. Aberer and M. Hauswirth, “Middleware support for the ‘Internet of Things’,” 2006.
10. A. Azzara, S. Bocchino, P. Pagano, G. Pellerano, and M. Petracca, “Middleware solutions in WSN: The IoT oriented approach in the ICSI project,” in Proc. 21st Int. Conf. Softw. Telecommun. Comput. Netw., 2013, pp. 1–6.
11. V. Issarny et al., “Service-oriented middleware for the future Internet: State of the art and research directions,” J. Internet Serv. Appl., vol. 2, no. 1, pp. 23–45, 2011.
12. S. Bandyopadhyay, M. Sengupta, S. Maiti, and S. Dutta, “A survey of middleware for Internet of Things,” in Recent Trends in Wireless and Mobile Networks. New York, NY, USA: Springer, 2011, vol. 162, pp. 288–296.
13. F. C. Delicato, P. F. Pires, and T. Batista, Middleware Solutions for the Internet of Things. New York, NY, USA: Springer, 2013.
14. K. Aberer, M. Hauswirth, and A. Salehi, “The global sensor networks middleware for efficient and flexible deployment and interconnection of sensor networks,” Tech. Rep., 2006.
15. M. Musolesi, C. Mascolo, and S. Hailes, “EMMA: Epidemic messaging middleware for ad hoc networks,” Pers. Ubiq. Comput., vol. 10, no. 1, pp. 28–36, 2005.
16. C.-L. Fok, G.-C. Roman, and C. Lu, “Agilla: A mobile agent middleware for self-adaptive wireless sensor networks,” ACM Trans. Auton. Adapt. Syst. (TAAS), vol. 4, no. 3, p. 16, 2009.

17. J. R. Silva et al., “PRISMA: A publish-subscribe and resource-oriented middleware for wireless sensor networks,” in Proc. Adv. Int. Conf. Telecommun. (AICT’14), 2014, pp. 87–97.
18. D. Raggett, “The Web of Things: Challenges and opportunities,” IEEE Comput., vol. 48, no. 5, pp. 26–32, May 2015.
19. R. Want, B. N. Schilit, and S. Jenson, “Enabling the Internet of Things,”IEEE Comput., vol. 48, no. 1, pp. 28–35, Jan. 2015.
20. L. Baresi, L. Mottola, and S. Dustdar, “Building software for the Internet of Things,” IEEE Internet Comput., vol. 19, no. 2, pp. 6–8, Mar./Apr. 2015.
21. L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” Comput. Netw., vol. 54, no. 15, pp. 2787–2805, 2010.
22. L. Yao, Q. Z. Sheng, and S. Dustdar, “Web-based management of the Internet of Things,” IEEE Internet Comput., vol. 19, no. 4, pp. 60–67, Jul./Aug. 2015.
23. Y. Qin et al., “When things matter: A survey on data-centric Internet of Things,” J. Netw. Comput. Appl., vol. 64, pp. 137–153, Feb. 2016.
24. TE Commission. (2008). “Internet of Things in 2020,” [Online]. Available: http://www.caba.org/resources/Documents/IS-2008-93.pdf
25. M. Scott and R. Whitney. (2014). “The industrial Internet of Things,”[Online]. Available:http://www.mcrockcapital.com/uploads/1/0/9/6/10961847/mcrock_industrial_internet_of_things_report_2014.pdf
26. Moor Insights and Strategy. (2013). “Behaviorally segmenting the Internet of Things (IoT),” [Online]. Available:http://www.moorinsightsstrategy.com/wp-content/uploads/2013/10/Behaviorally-Segmenting-the-IoT-by-Moor-Insights-Strategy.pdf
27. C. P. Greg Gorbach and A. Chatha. (2014). “Planning for the industrial Internet of Things,” [Online]. Available: http://www.arcweb.com/ brochures/planning-for-the-industrial-internet-of-things.pdf
28. P. F. Harald Sundmaeker, P. Guillemin, and S. Woelfflé, Vision and Challenges for Realising the Internet of Things. Pub. Office EU, 2010 [Online]. Available: http://www.internet-of-thingsresearch.eu/pdf/IoT_Clusterbook_March_2010.pdf
29. V. Ovidiu and F. Peter, Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems, 2013.
30. IT Union, “ITU Internet report 2005: The Internet of Things,” 2005.

31. M. P. Papazoglou and D. Georgakopoulos, “Introduction: Service-oriented computing,” Commun. ACM, vol. 46, no. 10,pp. 24–28, Oct. 2003. [Online]. Available: http://doi.acm.org/10.1145/944217.944233
32. C. Perera, A. B. Zaslavsky, P. Christen, and D. Georgakopoulos, “Context aware computing for the Internet of Things: A survey,” IEEE Commun.Surveys Tuts., vol. 16, no. 1, pp. 414–454, May 2013.
33. V. Cristea, C. Dobre, and F. Pop, “Context-aware environments for the Internet of Things,” in Internet of Things and Inter-Cooperative Computational Technologies for Collective Intelligence. New York, NY, USA: Springer, 2013, vol. 460, pp. 25–49.
34. G. Kortuem, F. Kawsar, D. Fitton, and V. Sundramoorthy, “Smart objects as building blocks for the Internet of Things,” IEEE Internet Comput.,vol. 14, no. 1, pp. 44–51, Jan./Feb. 2010.
35. D. Kyriazis and T. Varvarigou, “Smart, autonomous and reliable Internet of Things,” Proc. Comput. Sci., vol. 21, pp. 442–448, 2013.
36. P. Banerjee et al., “Everything as a service: Powering the new information economy,” Computer, vol. 44, no. 3, pp. 36–43, Mar. 2011.
37. A. Zaslavsky, C. Perera, and D. Georgakopoulos, “Sensing as a service and big data,” 2013.
38. D. Tracey and C. Sreenan, “A holistic architecture for the Internet of Things, sensing services and big data,” in Proc. 13th IEEE/ACM Int.Symp. Cluster Cloud Grid Comput. (CCGrid), May 2013, pp. 546–553.
39. IoT Analytics. (2014). Why the Internet of Things Is Called Internet of Things: Definition, History, Disambiguation. [Online]. Available: https://iot-analytics.com/Internet-of-things-definition/
40. Tinydb [Online]. Available: http://telegraph.cs.berkeley.edu/tinydb/
41. R. Rouvoy et al., “Middleware support for self-adaptation in ubiquitous and service-oriented environments,” in Software Engineering for Self-Adaptive Systems. New York, NY, USA: Springer, 2009, pp. 164–182.
42. A. Katasonov, O. Kaykova, O. Khriyenko, S. Nikitin, and V. Y. Terziyan,“Smart semantic middleware for the Internet of Things,” in Proc.ICINCO-ICSO, 2008, pp. 169–178.
43. M. Bakillah, S. H. L. Liang, A. Zipf, and M. A. Mostafavi, “A dynamic and context-aware semantic mediation service for discovering and fusion of heterogeneous sensor data,” J. Spatial Inf. Sci., vol. 6, no. 1, pp. 155–185, 2013.

44. J. Gubbi, R. Buyya, S.Marusic, andM. Palaniswami, “Internet of Things: A vision, architectural elements, and future directions,” Future Gener. Comput. Syst., vol. 29, no. 7, pp. 1645–1660, 2013.
45. H. Wang, X. Zhou, X. Zhou, W. Liu, W. Li, and A. Bouguettaya, “Adaptive service composition based on reinforcement learning,” in Service-Oriented Computing. New York, NY, USA: Springer, 2010, pp. 92–107.
46. G. Bin, Z. Daqing, and W. Zhu, “Living with Internet of Things: The emergence of embedded intelligence,” in Proc. 4th Int. Conf. Cyber Phys. Soc. Comput. Internet Things (iThings/CPSCom), Oct. 2011, pp. 297–304.
47. K. Modukuri, S. Hariri, N. V. Chalfoun, and M. Yousif, “Autonomous middleware framework for sensor networks,” in Proc. Int. Conf. Pervasive Serv. (ICPS’05), 2005, pp. 17–26.
48. R. Meier and V. Cahill, “Steam: Event-based middleware for wireless ad hoc networks,” in Proc. 22nd Int. Conf. Distrib. Comput. Syst.Workshops, 2002, pp. 639–644.
49. P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, “The many faces of publish/subscribe,” ACM Comput. Surveys, vol. 35, no. 2, pp. 114–131, 2003.
50. V. Issarny, M. Caporuscio, and N. Georgantas, “A perspective on the future of middleware-based software engineering,” in Proc. Future Softw. Eng., 2007, pp. 244–258.
51. M. Papazoglou, “Service-oriented computing: Concepts, characteristics and directions,” in Proc. 4th Int. Conf. Web Inf. Syst. Eng. (WISE’03), Dec. 2003 pp. 3–12.
52. D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio, “Interacting with the SOA-based Internet of Things: Discovery, query, selection, and on-demand provisioning of web services,” IEEE Trans. Serv. Comput., vol. 3, no. 3, pp. 223–235, Jul. 2010.
53. M. Caporuscio, P. G. Raverdy, and V. Issarny, “UbiSOAP: A service oriented middleware for ubiquitous networking,” IEEE Trans. Serv. Comput., vol. 5, no. 1, pp. 86–98, Jan./Mar. 2012.
54. C. L. Fok, G. C. Roman, and C. Lu, “Servilla: A flexible service provisioning middleware for heterogeneous sensor networks,” Sci. Comput. Programm., vol. 77, no. 6, pp. 663–684, 2012.
55. N. Reijers, K.-J. Lin, Y.-C.Wang, C.-S. Shih, and J. Y. Hsu, “Design of an intelligent middleware for flexible sensor configuration inM2M systems,” in Proc. SENSORNETS, 2013, pp. 41–46.

56. S. Hachem, “Service-oriented middleware for the large-scale mobile Internet of Things,” M.S. theses, Université de Versailles-Saint Quentin en Yvelines, Versailles, France, Feb. 2014.
57. Carriots [Online]. Available: https://www.carriots.com/
58. Xively [Online]. Available: https://xively.com/
59. N. Costa, A. Pereira, and C. Serodio, “Virtual machines applied to WSN’s: The state-of-the-art and classification,” in Proc. Int. Conf. Syst. Netw. Commun. (ICSNC’07), Aug. 2007, p. 50.
60. I. Marques, J. Ronan, and N. Rosa, “A register-based virtual machine for wireless sensor networks,” in Proc. Sensors, 2009, pp. 1423–1426.
61. P. A. Levis, “Application specific virtual machines: Operating system support for user-level sensornet programming,” Ph.D. dissertation, Berkeley, CA, USA, 2005.
62. J. Ellul and K. Martinez, “Run-time compilation of bytecode in sensor networks,” in Proc. 4th Int. Conf. Sensor Technol. Appl. (SENSORCOMM), Jul. 2010, pp. 133–138.
63. Z. Song, M. Lazarescu, R. Tomasi, L. Lavagno, and M. Spirito, “High level Internet of Things applications development using wireless sensor networks,” in Internet of Things, 2014, vol. 9, pp. 75–109.
64. M. Mamei and F. Zambonelli, Field-Based Coordination for Pervasive Multiagent Systems. Berlin, Germany: Springer-Verlag, 2005.
65. D. B. Lange and M. Oshima, “Seven good reasons for mobile agents,” Commun. ACM, vol. 42, no. 3, pp. 88–89, Mar. 1999.
66. R. Lopes, F. Assis, and C. Montez, “MASPOT: A mobile agent system for sun spot,” in Proc. Int. Symp. Auton. Decentralized Syst. (ISADS), Mar. 2011, pp. 25–31.
67. F. Aiello, G. Fortino, and A. Guerrieri, “Using mobile agents as enabling technology for wireless sensor networks,” in Proc. SENSORCOMM, Aug. 2008.
68. J. Ceclio and P. Furtado, “Existing middleware solutions for wireless sensor networks,” in Proc. Wireless Sens. Heterogen. Netw. Syst., 2014, pp. 39–59.
69. N. R. Jennings, “An agent-based approach for building complex software systems,” Communications, pp. 35–41, 2001.
70. E. Pignaton de Freitas, “A survey on adaptable middleware for wireless sensor networks,” 2008.
71. L. Mottola, A. L. Murphy, and G. P. Picco, “Pervasive games in a moteenabled virtual world using tuple space middleware,” in Proc. 5th ACM SIGCOMM, 2006, p. 29.

72. S. Hadim and N. Mohamed, “Middleware: Middleware challenges and approaches for wireless sensor networks,” IEEE Distrib. Syst. Online, vol. 7, no. 3, p. 1, Mar. 2006.
73. S. Sicari, A. Rizzardi, L. Grieco, and A. Coen-Porisini, “Security, privacy and trust in Internet of Things: The road ahead,” Comput. Netw., vol. 76, pp. 146–164, Jan. 2015.
74. M. Palattella et al., “Standardized protocol stack for the Internet of (Important) things,” IEEE Commun. Surveys Tuts., vol. 15, no. 3, pp. 1389–1406, 2013.
75. B. Shanmuga Sundaram, “A quantitative analysis of 802.11ah wireless standard,” International Journal of Latest Research in Engineering and Technology, vol. 2, 2016. View at Google Scholar
76. W. Sun, M. Choi, and S. Choi, “Ieee 802.11 ah: a long range 802.11 wlan at sub 1 ghz,” Journal of ICT Standardization, vol. 1, no. 1, pp. 83–108, 2013. View at Google Scholar
77. Jazib Frahim, Carlos Pignataro, Jeff Apcar and Monique Morrow, “Securing the Internet of Things: A Proposed Framework” [Online]. Available: https://www.cisco.com/c/en/us/about/security-center/secure-iot-proposed-framework.html#4
78. Pallavi Sethi and Smruti R. Sarangi, “Internet of Things: Architectures, Protocols, and Applications,” Journal of Electrical and Computer Engineering Volume 2017 (2017), Article ID 9324035, 25 pages. Available: https://www.hindawi.com/journals/jece/2017/9324035/