May 2017

Implementation and Analysis of Communication Protocols in Internet of Things

Priyanka Thota
University of Nevada, Las Vegas, priyankathota4@outlook.com

IMPLEMENTATION AND ANALYSIS OF COMMUNICATION PROTOCOLS IN INTERNET OF THINGS

By

Priyanka Thota

Bachelor of Technology, Computer Science
RGUKT IIIT-Basar, India
2015

A thesis submitted in partial fulfillment
Of the requirements for the

Master of Science in Computer Science

Department of Computer Science
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas
May 2017

Thesis Approval

The Graduate College
The University of Nevada, Las Vegas

March 17, 2017

This thesis prepared by

Priyanka Thota

entitled

Implementation and Analysis of Communication Protocols in Internet of Things

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science
Department of Computer Science

Yoohwan Kim, Ph.D.
Examination Committee Chair

Ajoy K. Datta, Ph.D.
Examination Committee Member

Juyeon Jo, Ph.D.
Examination Committee Member

Venkatesan Muthukumar, Ph.D.
Graduate College Faculty Representative

Kathryn Hausbeck Korgan, Ph.D.
Graduate College Interim Dean

ABSTRACT

IMPLEMENTATION AND ANALYSIS OF COMMUNICATION PROTOCOLS IN INTERNET OF THINGS

By

Priyanka Thota

Dr. Yoohwan Kim, Examination Committee Chair

Associate Professor, Department of Computer Science

University of Nevada, Las Vegas

Internet of Things (IoT) is the future of all the present-day devices around the globe. Giving them internet connectivity makes IoT the next frontier of technology. Possibilities are limitless as the devices communicate and interact with each other which make it even more interesting for the global markets. For example, Rolls-Royce announced that it would use the Microsoft Azure IoT suite and also the Intelligence suite of Cortana to keep track of the fuel usage, for performance analysis, to optimize the fly routes etc. which improves the airline efficiency. The devices must communicate with each other, the data from these devices must be collected by the servers, and the data is then analyzed or provided to the people. For all this to happen, there is a need for efficient protocols to ensure that the communication is secure and to avoid loss of data. This research is about the implementation and analysis of various protocols that can be used for the communication in IoT. Various protocols with various capabilities are required for different environments. The internet today supports hundreds of protocols from which choosing the best would be a great challenge. But each protocol is different in its own way when we have the specifics like security, reliability, range of communication etc. This research emphasizes on the best available protocols and the environments that suit them the most. It provides an implementation of some of the protocols and analyzes the protocols according to the results obtained. The data collected from the sensors/devices through a protocol is also subject to predictive analysis which improves the scope of the project to performing data analysis on the data collected through IoT.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor, Dr. Yoohwan Kim, for all the constant support and guidance throughout the course of my graduate studies and thesis at the University of Nevada, Las Vegas. His valuable suggestions and encouragement gave me the motivation to complete my thesis.

I am grateful to Dr. Ajoy Kumar Datta, who helped me in getting a Graduate Assistantship in the Department of Computer Science, guided me in every step of academic life for the past two years. I would also like to thank Dr. Juyeon Jo, Dr. Venkatesan Muthukumar and Dr. Ajoy Kumar Datta for being a part of my thesis committee and reviewing my thesis.

My deep sense of gratitude to my parents Annavara Satya Prasad Thota, Vani Thota and my brother Sai Subramayam Thota who are my moral strength and inspiration. I would like to thank Sai Ram Ganti for his constant guidance and encouragement throughout my years of study. Thank you to all my friends who have been my extended family in the United States and being there for me whenever I wanted someone to count on.

PRIYANKA THOTA

University of Nevada, Las Vegas

May 2017

TABLE OF CONTENTS

ABSTRACT iii

ACKNOWLEDGMENTS iv

TABLE OF CONTENTS v

LIST OF TABLES viii

LIST OF FIGURES ix

CHAPTER 1 - INTRODUCTION 1

1.1 APPLICATION 3

1.2 MANAGEMENT 4

1.3 SECURITY 4

1.4 IoT SERVICE 4

1.5 COMMUNICATION 5

1.6 DEVICES 5

1.7 TYPES OF IoT SYSTEMS 5

CHAPTER 2 - ARCHITECTURE OF INTERNET OF THINGS 7

2.1 CONSTRAINED APPLICATION PROTOCOL (CoAP): 9

2.2 EXTENSIBLE MESSAGING AND PRESENCE PROTOCOL (XMPP): 9

2.3 RESTful HYPERTEXT TRANSFER PROTOCOL (HTTP): 9

2.4 MESSAGE QUEUING TELEMETRY TRANSPORT (MQTT): 10

2.5 GOAL OF THIS RESEARCH 11

CHAPTER 3 - COMMUNICATION PROTOCOLS 13

3.1 MESSAGE QUEUING TELEMETRY TRANSPORT 16

3.1.1 Information Independent Nodes 16

3.1.2 Time Decoupling 16

3.1.3 Synchronization decoupling 16

3.1.4 Security Analysis 17

3.1.5 MQTT Quality of Service levels 17

3.1.5.1 Level 0: 17

3.1.5.2 Level 1: 17

3.1.5.3 Level 2: 18

3.2 CONSTRAINED APPLICATION PROTOCOL(CoAP) 19

3.2.1 User Datagram Protocol (UDP) 19

3.2.2 Send multiple nodes 19

3.2.3 Resource/Service Discovery 20

3.2.4 Async-Data Transfer 20

3.3 Wi-Fi 21

3.4 BLUETOOTH 21

3.5 MAJOR CLASSIFICATION OF PROTOCOLS 22

3.5.1 Wired/Network protocols [34] 22

3.5.2 TCP and UDP 23

CHAPTER 4 - EXPERIMENT AND ANALYSIS OF PROTOCOLS 24

4.1 WEB SERVER RUNNING ON ESP8266 24

4.2 TEMPERATURE AND HUMIDITY LOGGER, REST API BASED COMMUNICATION WITH MQTT 27

4.2.1 Uploading Data 29

4.2.2 Data Retrieval/Pull 30

4.3 THINGSPEAK API 31

4.4 SECURITY ISSUES 33

4.5 MINIMAL CoAP SERVER 33

4.6 COMMUNICATION VIA BLUETOOTH USING HC-04 AND ARDUINO 34

CHAPTER 5 - PREDICTION AND DATA ANALYSIS USING RATTLE 37

5.1 PREDICTION ANALYSIS 37

5.2 DATA ANALYSIS 40

CHAPTER 6 - COMMUNICATION MODES, PARAMETERS AND THEIR LIMITATIONS 41

6.1 COMMUNICATION MODES 41

6.1.1 Satellite 41

6.1.2 Wi-Fi 41

6.1.3 Radio Frequency (RF) 42

6.1.4 Radio Frequency Identification (RFID) 42

6.1.5 Near Field Communication (NFC) 42

6.1.6 Bluetooth 43

6.2 COMMUNICATION PARAMETERS 44

6.2.1 Baud Rate/Data Speed 44

6.2.2 Equipment/Hardware Durability 44

6.2.3 Hardware Cost 44

6.2.4 Power Requirement ... 44

6.2.5 Distance/Range or communication ... 45

6.3 LIMITATIONS AND ADVANTAGES ... 45

6.3.1 Bluetooth ... 46

6.3.2 Wi-Fi ... 47

6.3.3 NFC ... 47

CHAPTER 7 - SECURITY ISSUES ... 48

7.1 VULNERABILITIES ... 48

7.2 DEFENSE ... 49

7.2.1 Secure booting ... 50

7.3 AUTHENTICATION, ENCRYPTION AND INTEGRITY ... 52

7.3.1 Authentication ... 52

7.3.2 Encryption ... 53

7.3.3 Integrity ... 53

CHAPTER 8 - CONCLUSION ... 54

CHAPTER 9 - FUTURE WORK ... 56

BIBLIOGRAPHY ... 58

CURRICULUM VITAE ... 62

LIST OF TABLES

Table 1: Comparison of communication protocols for IoT devices 10

Table 2: Quick Comparison of MQTT and CoAP.. 15

Table 3: Wi-Fi Protocols and their Data rates 21

Table 4: Protocols and their vulnerabilities 49

LIST OF FIGURES

Figure 1: Range of IoT hardware development boards available with multiple features. 2

Figure 2: Architecture for a minimal IoT system.......... 3

Figure 3: Internet of Things communication network 7

Figure 4: Statistics of the number of connected IoT devices.......... 8

Figure 5: M2M Architecture.......... 14

Figure 6: Levels of QoS.......... 18

Figure 7: Comparison of MQTT and CoAP 21

Figure 8: Testing ESP8266 by running a web server on it. 24

Figure 9: Web server running on ESP8266 can be accessed anywhere on the network. 25

Figure 10: Pin Out diagram for ESP8266 Development Board 26

Figure 11: LUA Script to run a simple Web Server within local network.......... 26

Figure 12: Raspberry-Pi with a DHT11 connected over I2C.......... 27

Figure 13: Python script to push data to Thingspeak 28

Figure 14: Thingspeak data visualization 29

Figure 15: Code Snippet for Thingspeak API.......... 29

Figure 16: Dynamic representations add-ins for Thingspeak 30

Figure 17: Thingspeak role in IoT network.......... 32

Figure 18: CoAP Server running on Raspberry-pi. 33

Figure 19: CoAP client running on a remote computer. 34

Figure 20: Arduino and HC-04 interaction via Serial Port.......... 35

Figure 21: Support and confidence for a given dataset.......... 38

Figure 22: Predicted Values and Leverage plot 39

Figure 23: Proper Design principles for IoT.......... 50

CHAPTER 1

INTRODUCTION

Internet of things has been evolving over time when the need of connectivity was realized. Various architectures of IoT were designed according to the requirements of the system. The main idea of IoT was the flow of data between devices and ease of communication across a large number of nodes. In the very beginning communication was a barrier since most of the devices and technologies were still evolving and limited to wired communications [1]. Large distance communications were costly to implement and were less robust. Later wireless technologies like Bluetooth, Wi-Fi made the communication very fast and easy to implement. As the complexity and size of the system continued to increase, different aspects of security issues and management issues were noticed. All this development had to be standardized eventually to create a platform for all the industries, such that every product would be compatible with other products even in a different environment. Hardware was substantially produced by giant companies like Cisco, Intel, IBM etc. [2] and all these hardware nodes or equipment's individually followed a standardized architecture.

No sooner once the hardware was standardized in terms of standard connectivity (Ports, power requirements) software applications began to harness the power of these developments. The mobile market had widely changed and an average cellular device could run an application and communicate to any server. This greatly revolutionized the way hardware could interact. What had to be a physical button on a device was eventually replaced by a software touch/click within the application and in turn saving lots of money on hardware and replacement/maintenance costs. Below in Figure-3, we can see the latest hardware as of 2017 which are available for development of IoT applications. All of these development boards listed below are equipped with Wi-Fi, Bluetooth and some with NFC, IR, and an Ethernet port as well. These modules are stackable to create a hub of multiple devices and increase the processing power.

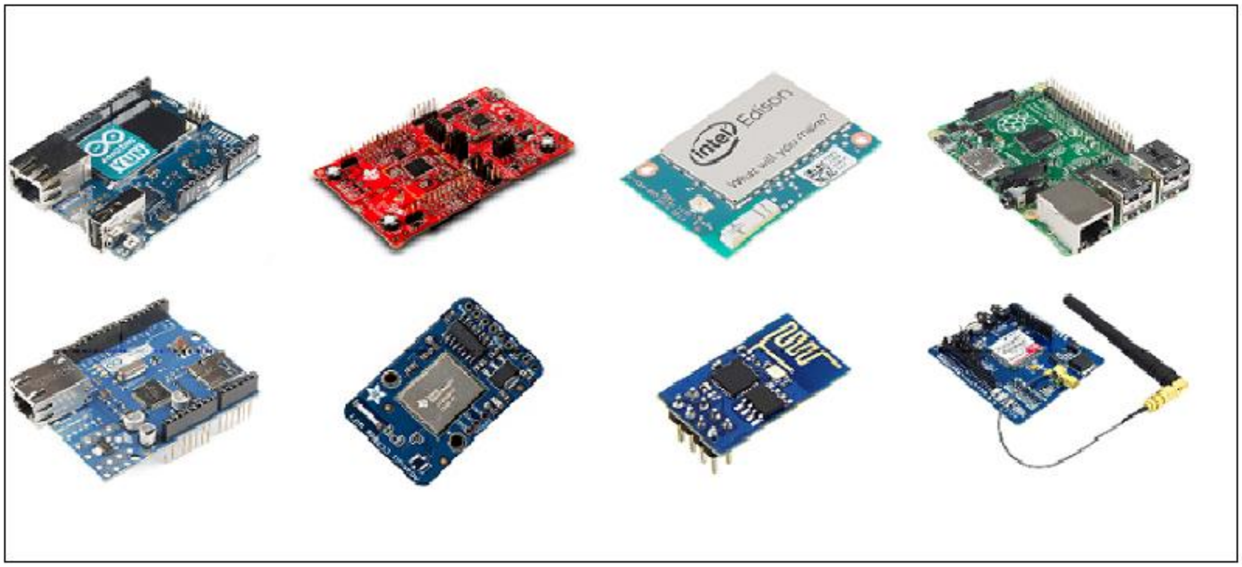


Figure 1: Range of IoT hardware development boards available with multiple features.

The architecture was re-designed multiple times along with different considerations for every iteration. As mentioned IoT was evolving for industries of various sizes and domains and a single architecture would not be sufficient to cater the needs of everyone. For large industries where data precision is crucial, timing and security are important, a much complex design was needed. Where else for a small industry with hardly 50+ connected nodes, reliability was more important than high-end security. This brought up a wide classification of architecture for IoT [3].

Since most of the IoT's communication happens over the wireless technologies/medium, security stands as a serious issue for big companies which incorporate IoT into their systems. For example, a home automation company may maintain a central server connected to 1500+ homes in a given area, with each home housing at least 50+ individual sensors/nodes to provide relevant data. Each of this home and its sensors are controlled by an application whose administrator would be the home owner himself and the main control privileges would lie with the company providing the service. Any issue of security or data breaching [4] into the servers of this company can instantly affect the homes located in its controls. This can be very serious since by control we mean almost anything can be done right from power outage to malfunctioning of HVAC systems in a home.

Below is an architecture representation for any minimal IoT system. This illustrates different modules within an IoT system and the way they communicate in between. This architecture shows the components involved in an end to end IoT system, starting from an application to the final device to be controlled or receive data.

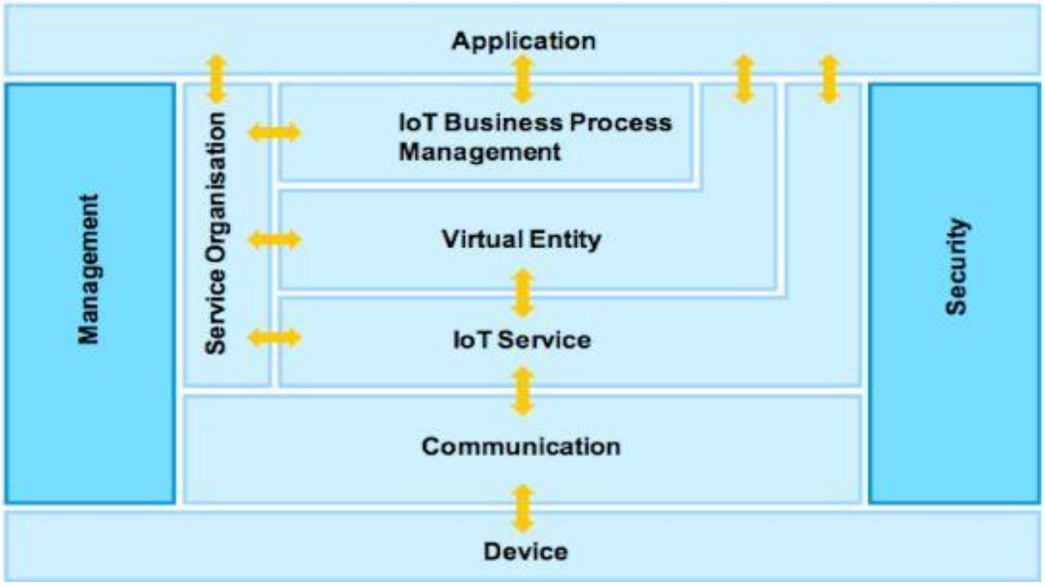


Figure 2: Architecture for a minimal IoT system

1.1 APPLICATION

This is the user side piece of software which could be running on any device/platform or on any operating system. This could be full-fledged software applications which may run on a computer with a matured operating system or a semi-functional android/IOS application [5] which runs on the mobile platform. This is the gateway through which the user would like to get or publish data to any service he is subscribed to. Essentially the application will choose/command the end hardware on when to collect data or how to respond to a given scenario. An application will have its own layer of security like user authentication via password or a fingerprint reader. These will natively communicate via Wi-Fi/Bluetooth or Ethernet depending on the device they are installed and will send/ receive from the end device. The application will only be responsible for sending data securely and verify the received data. It may use various encryption [6] methods in this process.

1.2 MANAGEMENT

This is one of the crucial components for an IoT system. It could be a piece of software maintained by the system itself or an individual. What it does is make sure the transactions happen safe and securely irrespective of the safety measures taken at different layers. A management component may handle various consumer/customer issues like error reporting or malfunctions in a system [7]. It acts as a portal to log complaints and improve the user experience for that system.

1.3 SECURITY

It plays a vital role and makes the whole application reliable and dependable. The complexity of this module is dependent on the size and requirement of the system. A big system with thousands of nodes may implement a lengthy encryption/decryption system to make sure data integrity is not lost. This module is not independent and is applicable to every individual module listed in the above diagram. Security is required at the application layer, transport layer and at the hardware end as well. Any compromise at any layer will be a threat to data and the system itself [8].

1.4 IoT SERVICE

This module is the core of the system operation. It is responsible for providing different services to users. Not necessarily only industries use IoT, but there can be individuals who may want a service to connect only two or three nodes in their home and may want a cloud service to cache their sensor data and use it for personal use. In such cases, for example, we have Amazon AWS [9] services which provide a wide variety of services targeted at different customers. These services can reduce the complexity of an IoT system in user perspective by a lot. All the user has to do is to buy a compatible IoT module and subscribe to the service provided and he/she can use the system right away. This reduces the overall cost for the user since he only needs to buy a service and not the whole system.

1.5 COMMUNICATION

This has set the standards for how modules from different manufacturers will communicate. It got the most robust communication protocols to be set as the standards like MQTT, CoAP, and HTTP [10] etc. A standardized protocol will mean ease of communication irrespective of the specifications of any device. Communication protocols again are widely divided into categories based on the pay load, range and security.

1.6 DEVICES

These are the end components of any IoT system. The entire system runs to control these end devices timely as and when required. These devices get upgrades frequently and hence the system has to be on par to make sure the newly replaced device is compatible. For example, a set of light bulbs in a home can be controlled using a central server linked to a smartphone [11]. In unlikely case, if a bulb is damaged or replaced with a new one, still the system should remain unaffected and work seamlessly.

1.7 TYPES OF IoT SYSTEMS

There is no actual classification for different IoT systems, but considering the size and environment of implementations they can be widely classified into the following categories:

1. Large Industries
2. Integrated systems for small areas
3. Personal systems
4. Wide Area Services

Large Industries use up to few thousands of sensors to measure wear and tear on the machinery, to get counts of different items, to get the status of a car which is on the manufacturing belt etc. They may use wired systems for better reliability [12] and easy troubleshooting. These systems may not necessarily connect to the internet but will be communicating to a local server within the intranet. Integrated systems for small areas include intelligent farming with few hundreds of sensors sending humidity data to a water pump which turns on when required. Personal systems will include a home security monitoring system integrated with appliance control. Lastly, wide area services will include energy meters spread over large areas.

CHAPTER 2

ARCHITECTURE OF INTERNET OF THINGS

Internet of Things (IoT) refers to the interrelated objects, people, animals, devices, mechanical machines etc. that are provided with unique identifiers for internet connectivity and to provide them with an interaction medium to transfer data over the network without any human involvement. It is mainly about Machine-to-Machine (M2M) [13] communication which is mainly built on cloud computing and a network of sensors that gather data.

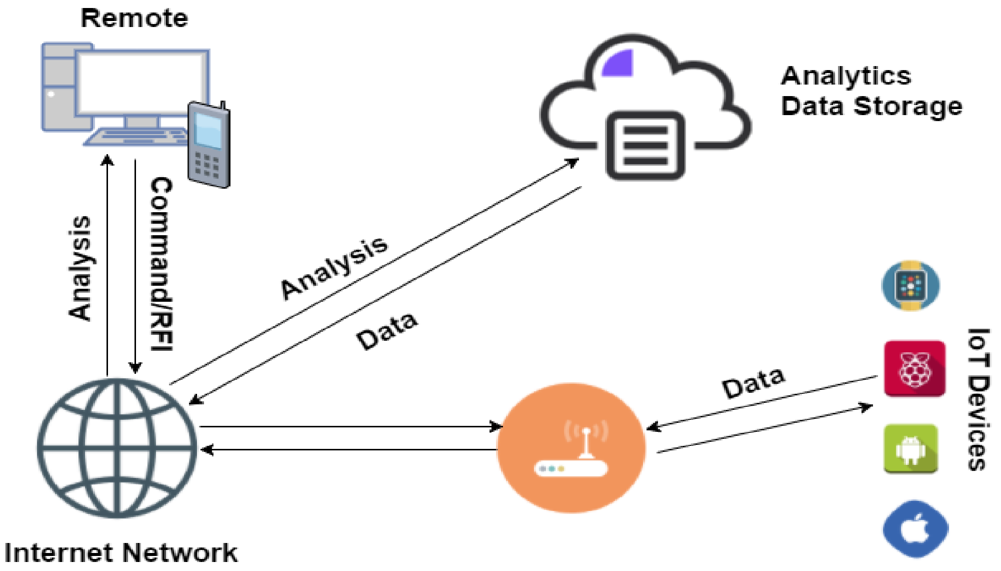


Figure 3: Internet of Things communication network

Some examples of the devices that can fall into the scope of Internet of Things are a thermostat, an animal with biochip transponder, a person with heart monitor implant etc. Security plays a major role in the maintenance of an IoT system. Considering an example of the motion sensors that turns the lights on or off and the temperature sensors that control the Heating, Ventilation and Air Conditioning (HVAC) systems the damage done through intrusion might be small like if someone hacks into the motion sensor the light might stay on forever. On the other hand, self-driving cars use IoT systems. If someone could hack into the server even a small change or loss of data could be catastrophic. IoT is going to make everything in our lives from bulbs to airplanes "smart" [14]. Currently, there are around 12 billion devices that can connect to the internet. The number of IoT devices is growing around the globe in leaps and bounds. Researchers estimate that the number of IoT devices will exceed 50 billion by 2020.

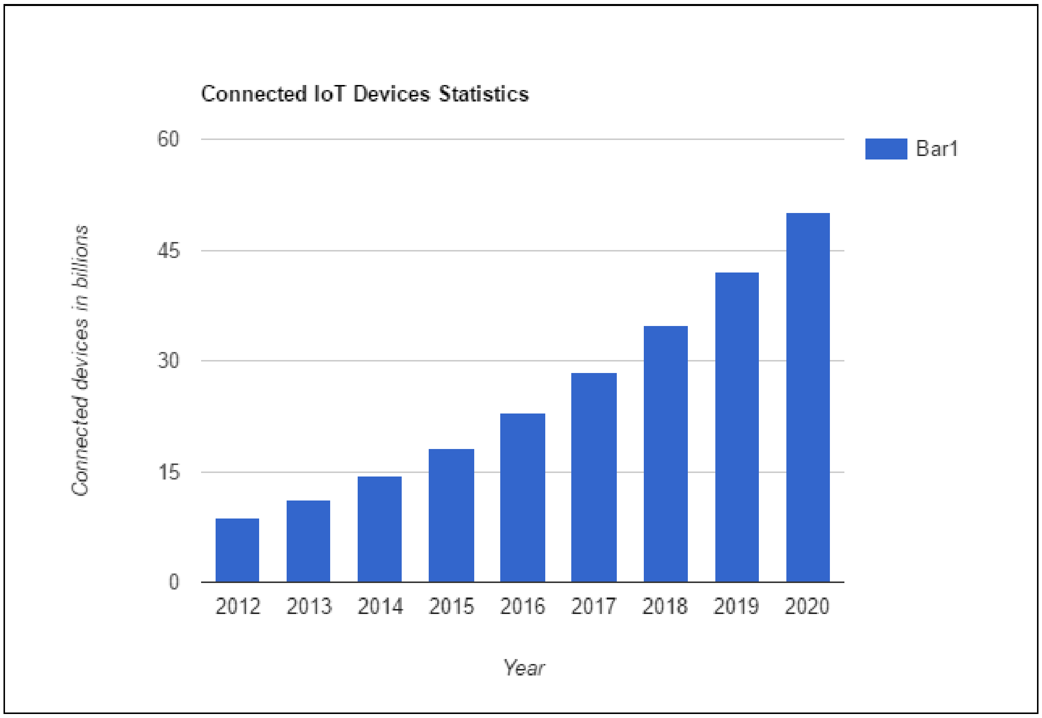


Figure 4: Statistics of the number of connected IoT devices

One of the areas in IoT where security can be improved is by choosing the right communication protocols. Communication protocols are a set of digital rules that are required to exchange data among computing devices. The devices on the network have to agree upon various aspects of data exchange for successful transmission to happen. Today there are hundreds of protocols that are supported by the internet. Using the right protocol according to the environment and the devices being used has a significant importance. This could improve the security of IoT system by a consistent level. Some of the important and widely used IoT protocols [15] are described below:

1. Constrain Application Protocol (CoAP)
2. Extensible Messaging and Presence Protocol (XMPP)

3. Hyper Text Transfer Protocol (HTTP)
4. Message Queuing Telemetry Transport (MQTT)

2.1 CONSTRAINED APPLICATION PROTOCOL (CoAP):

CoAP is mainly a web transfer protocol used for constrained devices and networks. It has strong security built into the protocol using Datagram Transport Layer Security (DTLS). It is a completely asynchronous protocol. CoAP is a very efficient RESTful protocol specialized for M2M applications. It is easy to proxy to/from HTTP. CoAP is not a general replacement for the HTTP protocol. This protocol is designed to the minimum resources on the network and devices being used. Some features of CoAP include User Datagram Protocol (UDP) binding with reliability and multicast support, GET, POST, PUT, DELETE [16] methods, URI support, and built-in discovery.

2.2 EXTENSIBLE MESSAGING AND PRESENCE PROTOCOL (XMPP):

The primary method of communication using XMPP is short messages. It can be customized for individual needs like the "jingle" extension can be used for audio and video purposes. It is reactive to the user's presence and status i.e the presence indicator determines the state of an XMPP entity. XMPP takes place inside a single data stream [17]. It is basically a technology used on the network to stream XML. Any person over the network can create his/her own XMPP server, it is similar to that of an email. This helps the organizations and also individuals to create their own communication experience. XMPP mainly is for instant messaging and detecting the online presence.

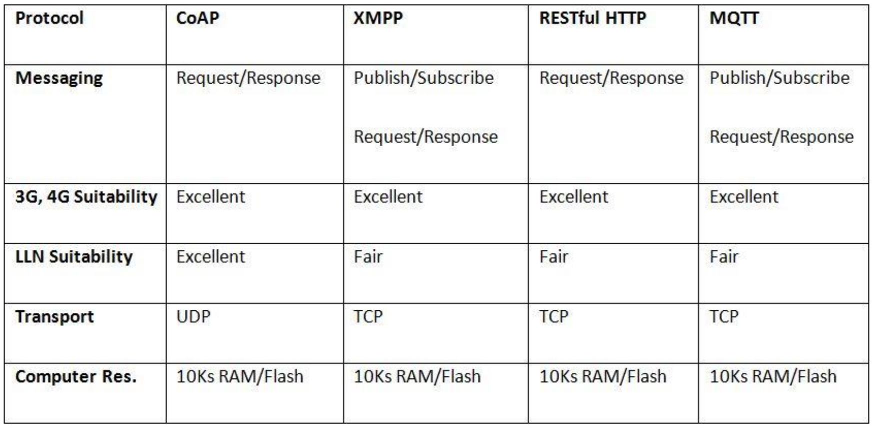
2.3 RESTful HYPERTEXT TRANSFER PROTOCOL (HTTP):

REST stands for Representational State transfer. HTTP is a basic protocol used for communication in web related services. REST is a way to use HTTP which is an application layer protocol. The REST set of rules make HTTP self-descriptive and gives HTTP a hypermedia constraint that is Hypermedia as the Engine of Application State (HATEOAS). Self-descriptive is when the request is to be completely self-descriptive in the intent of the users. HATEOS turns the application into a series of web links and the current state of the client is based in the place in the web. REST architecture does not allow us to use session context while communicating with the client. REST supports various data formats [18].

2.4 MESSAGE QUEUING TELEMETRY TRANSPORT (MQTT):

MQTT is a protocol designed to gather data from the devices and communicate it to the servers. It is a lightweight protocol using the publish/subscribe mechanism. The name suggests that this protocol is for remote monitoring. MQTT [19] works on Transmission Control Protocol (TCP) to ensure reliability and prevent data loss. It is a good choice for the networks experiencing various levels of latency due to poor bandwidth. If the connection between the broker and subscribing client is broken, the broker buffers the messages and sends them to the subscribing client when it is available. If the connection gets terminated between the broken and publishing client, the broker can then close the connection and send a cached message to the subscribing clients with the instructions from the publishing client.

Depending on the evolving hardware, new protocols have been coming up. Old protocols may not yield complete throughput on a new hardware. Just like how a new operating system may run slower on old hardware.



Protocol	**CoAP**	**XMPP**	**RESTful HTTP**	**MQTT**
Messaging	Request/Response	Publish/Subscribe Request/Response	Request/Response	Publish/Subscribe Request/Response
3G, 4G Suitability	Excellent	Excellent	Excellent	Excellent
LLN Suitability	Excellent	Fair	Fair	Fair
Transport	UDP	TCP	TCP	TCP
Computer Res.	10Ks RAM/Flash	10Ks RAM/Flash	10Ks RAM/Flash	10Ks RAM/Flash

Table 1: Comparison of communication protocols for IoT devices

All the above-mentioned protocols have a wide range of implementations for different requirements in IoT. A deeper understanding of the protocols and the application requirements is necessary to choose the best protocol that is suitable for the application.

2.5 GOAL OF THIS RESEARCH

This research aims at the analysis of various communication protocols in the IoT environment. There are a huge number of manufacturers for IoT hardware with various levels of standards. To meet certain common goals and standards, some protocols have been developed. These protocols are developed with different constraints into consideration which will be discussed in further chapters. This research extensively focused on protocols, their advantages and the type of environments suitable for them. It gives a better understanding for anyone on how to choose between various modes of communication whether it could be hardware or a simple protocol. Specific experiments were conducted to analyze the most used protocols and then further data analysis was performed on collected IoT data. This research exposed some of the latest hardware developments in the field of IoT such as nodeMCU [20] which has an esp8266 controller. These hardware chips are not yet popular, but this research work will throw some light on to low-cost IoT development and design.

Multiple projects along with academic work were combined to make this research more effective. Previously some work has been done on IoT security which analyzed the security loop holes at various layers of transportation. These experiments were executed on full-fledged IoT development hardware like Raspberry-Pi and Beagle-Bone-Black. This research utilized extremely low-cost hardware like the esp8266 and the cheaper models of Raspberry-Pi [21]. It has been challenging to determine what kind of setup and environment does a given IoT system need. Everything from protocols, hardware selection and modes of communication need to be properly chosen to create an effective IoT system. This work would simplify setting up an IoT system in all aspects. Possible security issues also have been discussed in detail. The main motivation for this work was the interaction with simple systems everyday which had different IoT architectures in the background. This resulted in working more and getting a deeper understanding of how IoT systems are integrated and why not all IoT systems can be same in terms of protocols, communication modes and the hardware’s used.

CHAPTER 3

COMMUNICATION PROTOCOLS

There are many customized protocols that exist for the IoT to be implemented. But only a handful of them has been standardized for all the devices which work on IoT. The Wireless Sensor Networks are being implemented in large scale to monitor different parameters like temperature humidity etc. In order to ensure whether the message/packet is delivered or not, CoAP features a re-transmission mechanism, while MQTT relies on TCP. In remote areas which lack wired connectivity/network, the only available Internet access technology is wireless or satellite links, which involve significant losses and delays.

With this increase, the devices need to be classified based on various parameters like power consumption, requirement of bandwidth, form factor of the device itself. Each device will act as a node collecting data from environment and transmitting over the net to master nodes. Once can take an example of a soil humidity sensor connected and controlled through the web [22]. This sensor may have a small micro-controller and a battery source as it would be located remotely under soil for monitoring. This sensor node needs to send the values of humidity in regular intervals of time.

Now we get the need for a light-weight protocol for data transfer across the sensor node. We need a light-weight protocol not just to keep the bandwidth low but also to minimize power consumption and extend the sensor node life. In this paper, we will be discussing the prominent transfer protocols used in the current Internet of Things scenario. In the above-mentioned case there is no need of any user-interface to be loaded on the node, rather it can be handled after receiving at the master side. The protocols used for such node to node communications in IoT are known as M2M (Machine to Machine). Figure 5 shows the architecture of M2M applications and their protocols.

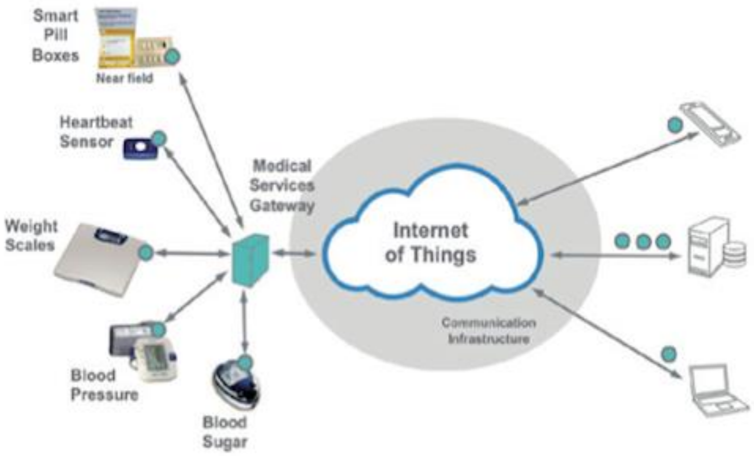
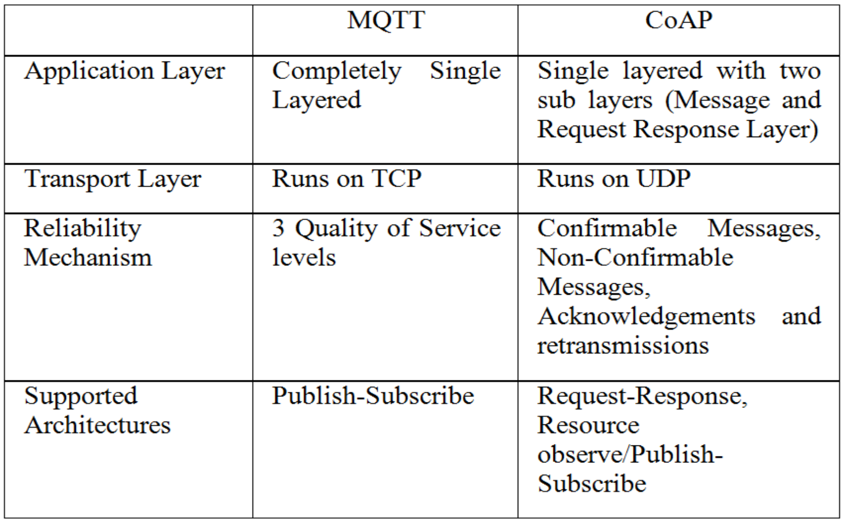


Figure 5: M2M Architecture

The M2M opportunity is dynamic and is growing fast. It is currently providing connectivity for variety of devices like sensors, mobile devices, health monitoring systems, utility meters [23] etc. This is because everyday objects which we are using are getting updated and can be addressed, recognized and controlled via networks. The two most evident protocols that will be analyzed in this research are as shown below. Each of these protocols will be analyzed tested and interpreted on IoT Development Board ESP8266.

1. MQTT (Message Queuing Telemetry Transport)
2. CoAP (Constrained Application Protocol)

Both of the above mentioned protocols are open standards and are better suited to constrained environments than HTTP. These protocols have wide range of implementations for different requirements in the IoT field. They both provide mechanisms for asynchronous communication and run on IP. There are other major protocols that are used in a smaller environment or as a base protocol like Wi-Fi, Bluetooth, Infra-Red, Zigbee [24] etc. But among these MQTT and CoAP are widely used because of their lightweight design and advantages.



	MQTT	CoAP
Application Layer	Completely Single Layered	Single layered with two sub layers (Message and Request Response Layer)
Transport Layer	Runs on TCP	Runs on UDP
Reliability Mechanism	3 Quality of Service levels	Confirmable Messages, Non-Confirmable Messages, Acknowledgements and retransmissions
Supported Architectures	Publish-Subscribe	Request-Response, Resource observe/Publish-Subscribe

Table 2: Quick Comparison of MQTT and CoAP

From the above table, the major difference between CoAP and MQTT is that the CoAP runs on the top of UDP (User Datagram Protocol) while the MQTT runs on TCP (Transfer Control Protocol). UDP is not reliable and CoAP provides its own mechanisms and functions to achieve the reliability mechanism. This is achieved by the use of “Confirmable Messages” and “Non-Confirmable Messages”. The former one requires an acknowledgment while the latter doesn’t need any. The quality of service levels is distinguished in MQTT where else CoAP does not provide any differentiated levels of [25] Quality of Service (QoS).

MQTT can be defined as multiple node communication protocol for communication between different nodes through a single node. Here the broker or the server will decide where to route and copy messages. CoAP is primarily a one to one protocol for sending state information between the clients and servers. CoAP is more suited to a state transfer model rather than purely event based. MQTT does not label its data or attach any kind of metadata. The clients need to have up-front details for a smooth communication. With major amount of sensor data being collected prominently using MQTT and CoAP, this analysis will be helpful. Other protocols like Bluetooth and Zigbee are range limited and hence are applicable to smaller networks or minimal environment with less than 50 nodes.

3.1 MESSAGE QUEUING TELEMETRY TRANSPORT

MQTT is Publish/Subscribe messaging protocol designed for lightweight M2M communications between multiple nodes. It was originally developed by IBM and now is an open standard. Every client connects to the server using Transmission Control Protocol (TCP) [26] which is connection oriented and much reliable. MQTT has a very small overhead to transport and the protocol exchanges are minimized to reduce the traffic in the network. The benefits of publish-subscribe models for MQTT are mentioned below:

3.1.1 Information Independent Nodes

Every sensor node and the subscribed node would need their respective addresses. Every module can send data and ask/request for any other node's published information without any data regarding each other just because entire data goes through the central server or the broker to whom they are connected to. This reduces the data overhead which in turn increases the number of TCP requests. This basically allows modules to operate on their own [27].

3.1.2 Time Decoupling

In the sensor network certain nodes may be active or in sleep mode at a given time. Sleep mode enables low power mode to be activated which reduces the power consumption during unwanted times greatly [28]. A node could now send information even if another node is sleeping. Other modules/nodes will receive their data back when they are available or wake up from sleep mode. The publisher and the subscriber do not need to run at the same time.

3.1.3 Synchronization decoupling

The messages are always stored in a queue by the broker/server until they are consumed by a client or node. If a node is processing with some published information and another node publishes new information to which the current node has subscribed, then the message is queued until the current operation is completed. This will minimize power usage and will reduce repeated operations on the devices that were sleeping.

3.1.4 Security Analysis

There is an extended version of MQTT known as SMQTT [29] where S stands for Secure. This uses encryption based lightweight protocol. This allows broadcast encryption, where one message is encrypted once and multiple clients receive that message securely. This process consists of four phases namely setup, encryption, publish and decryption. In the first phase the subscribers and the publishers register themselves to the broker and get the master secret key.

3.1.5 MQTT Quality of Service levels

In this, each level would indicate the responsibility of a message being delivered. All the three levels are described below:

3.1.5.1 Level 0:

This would send the message and also is called as send it and never acknowledge. It is just a one-time send without confirmation about the message reaching the destination. It is more suited to situations where the importance is low.

3.1.5.2 Level 1:

This would make sure the data reaches one-time minimum. It guarantees that the data reaches the destination at least once. It gives an acknowledgment about the message arrival at the intended destination. When the data being sent is received and confirmed by the other node, it will send a response/acknowledgment to the sending node. The sender waits for acknowledgment, and it saves the data to be sent and keeps resending it at regular intervals.

3.1.5.3 Level 2:

This rule will make sure that the data is received and properly fetched by the intended node. This is reliable level among the mentioned levels [30]. When the sender has QoS level 2 messages it sends a message announcement. The proposed receiver gets prompt and then would decrypt it. It shows that it is ready to receive data. The publishing node sends the data, and when the receiver decrypts and gets the data, the whole process is done with an ack/talk back. It is mainly useful for turning on or off alarms or lights in a house. Figure 6 shows the QoS levels for MQTT.

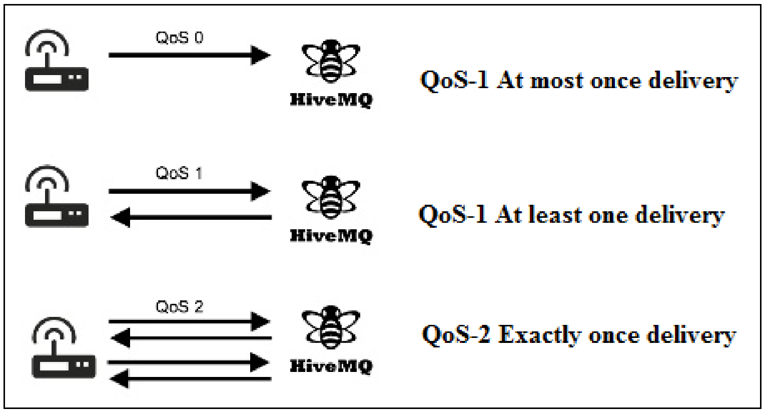

QoS 0
HiveMQ
QoS-1 At most once delivery
QoS 1
HiveMQ
QoS-1 At least one delivery
QoS 2
HiveMQ
QoS-2 Exactly once delivery

Figure 6: Levels of QoS

There are certain dis-advantages with MQTT which are as following:

1. Central Broker
2. TCP
3. Wake-up Time

For a distributed system, using central broker might be a disadvantage. This can become the single point failure for the entire system. TCP was designed to handle higher data loads and complex handshakes, In IoT this increases waking a node talking times and brings down battery life, if portable. As a result using TCP without session persistence, for data nodes with intermittent traffic can cause reduce the operating life of a given node.

3.2 CONSTRAINED APPLICATION PROTOCOL(CoAP)

This is another popular IoT protocol which gives one on one response/request with IETF standardization. CoAP is a transfer protocol for the documents. It works mostly for the constrained devices. It works on User Datagram Protocol (UDP) which is connectionless protocol. The advantages of CoAP are as follows:

3.2.1 User Datagram Protocol (UDP)

CoAP uses UDP as its background which is a bit less reliable. It also relies on redundant way of messaging rather than standard connections. Let us take an example of temperature sensor which may send a burst of data or one value at a time in a regular interval of time, even though nothing has changed from previous data values to new data values [31]. If a receptor on other end loses data for one iteration, the next chunk comes shortly and would make not much difference in terms of data content.

The packets here are connectionless which makes it unreliable but they allow quicker wake from sleep and data send/ack cycles. They also use lesser payload for packets which reduces the overall payload. It helps nodes to save power and operate for a long time.

3.2.2 Send multiple nodes

CoAP network is basically node to node. It also favors multi-node or single to multiple multi-transfer requirements. This is possible because constrained application protocol is basically built with IPV6 support which by default allows multiple node transfers. It would be less beneficial to send data to nodes which are in sleep mode, affects the power cycle.

3.2.3 Resource/Service Discovery

URI (Uniform Resource Identifier) is used by CoAP to implement a standard communication interface to given nodes in any network. The capabilities of the destination node are partially understood by its URI details and allow the packets to have additional flexibility. A line-powered flow-control actuator will have a different URI when compared to the one which is powered by a portable source. Data that is communicated with a battery powered device can be expected to be less reliable and repetitive since it has few limitations and same is not the case with a powered by wire device since it is more reliable and need not worry about its resources.

3.2.4 Async-Data Transfer

Many messages in the constrained application protocol are transmitted using request and report standard model. There are many other ways which will allow nodes to work more independently. For example, just like publish and subscribe which allows nodes to manage other nodes without much activity [32], CoAP also has a simplified "observe" mechanism.

Considering an example, the observing mode for node-1 can manage node-2 for a given data transfer cycle. Whenever a second node sends any relevant data, primary node will receive it after its sleep cycle. Any one of the node saves data for its observers. This might be similar to working mechanism when compared to MQTT. But it doesn't have any particular need of a broker, and need not hold any messages for other observers.

There are also certain issues with the CoAP. This protocol is less mature than MQTT in the current market. The reliability of CoAP is comparatively less with what MQTT provides in terms of levels of QoS. Figure 7 shows a quick comparison of architectures of MQTT and CoAP with respect to sensor nodes and clients.

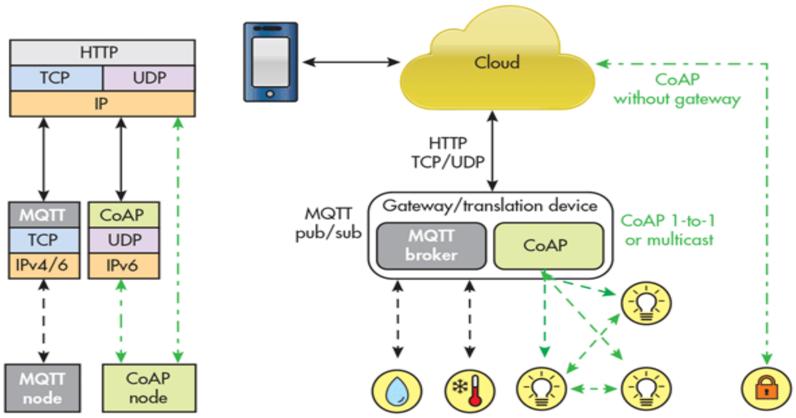


Figure 7: Comparison of MQTT and CoAP

3.3 Wi-Fi

Wi-fi is another very common wireless protocol which has multiple variants within itself. Each protocol can support different bandwidth; hence an appropriate one can be selected depending on the payload. The table below shows the available Wi-Fi protocols [33].

Protocol	Frequency	Signal	Maximum data rate
Legacy 802.11	2.4 GHz	FHSS or DSSS	2 Mbps
802.11a	5 GHz	OFDM	54 Mbps
802.11b	2.4 GHz	HR-DSSS	11 Mbps
802.11g	2.4 GHz	OFDM	54 Mbps
802.11n	2.4 or 5 GHz	OFDM	600 Mbps (theoretical)
802.11ac	5 GHz	256-QAM	1.3 Gbps

Table 3: Wi-Fi Protocols and their Data rates

This research is aimed at IoT-specific protocols like MQTT and CoAP; hence Wi-Fi and Zigbee are not detailed.

3.4 BLUETOOTH

Bluetooth is another protocol for communications over short ranges and is limited in terms of the number of devices connected to each other and bandwidth as well. It is a good fit for small scale IoT systems where the numbers of nodes are less than 10 and power consumption is not a considerable factor.

3.5 MAJOR CLASSIFICATION OF PROTOCOLS

We have discussed above regarding the important protocols which were used in experiments conducted during this research. This unit is about a general classification of protocols to which the above mentioned are a subset.

3.5.1 Wired/Network protocols [34]

- Ethernet

This is one of the widely used wired protocol and it uses certain methods namely carrier sense multiple access and collision detection. These methods are the core principles on which Ethernet works. Every system connected on Ethernet would check for traffic on the wire and if some other system is transmitting then this system would wait until its done and then put its data on the wire. Sometimes two systems may put data at the same time and collision may happen. During this time both systems will rollback their data and randomly fire data again. The collision delays are significantly small and do not affect the communication to and fro times.

- Local talk

This is quite similar in operation to Ethernet. This was developed by Apple and the only additional feature it has is any system before transmitting data over the wire would transmit a signal for intent. This will allow less collision of data between systems.

- Token Ring

This was developed by IBM and uses tokens for communication. In this protocol, all systems are connected in a logical ring, such that token moves from one system to other. If a system is idle and doesn't transmit any information, then the token moves around the ring. In another case the information/data to be transmitted will be attached to the token and will rotate around until reaches the required system [35].

- Asynchronous Transfer Mode

This mode can achieve a speed of 155Mbps and higher. Other protocols use different packet size or lengths for communication, but ATM uses fixed size packet transfer and can achieve high speeds over different local area networks.

- Gigabit Ethernet

This is one of the latest advancement for the Ethernet protocol which was discussed earlier. The transmission speed can reach up to 1 Gbps. In the future it will be used for workstations and servers.

3.5.2 TCP and UDP

Transmission Control Protocol and User Datagram protocol are two types of Internet protocols. TCP is a connection-oriented protocol and UDP is a connectionless protocol where data is sent in small packets in chunks. TCP is significantly slower than UDP because of its weight. In TCP there is always absolute guarantee that data is transferred if not attempts are made to re-transfer or recover a lost data packet. It is not so in UDP, faulty or lost packets are dropped and no attempts are made to re-transmit them. UDP is more used where fast and efficient transmission is required.

CHAPTER 4

EXPERIMENT AND ANALYSIS OF PROTOCOLS: CoAP, REST API BASED, MQTT AND BLUETOOTH

As part of this research, some of the most applied and frequently used protocols were selected for generating performance benchmark and analysis reports. The most commonly used IoT protocols as of 2017 are MQTT, CoAP and REST API-based communication and this chapter contains some experiments that were conducted specific to these protocols. Hardware was chosen accordingly to fit the needs of these protocols and various kinds of tests were conducted like a load test with multiple connection requests, testing in different environments and analyzing the flexibility of each protocol and its behavior to different cases.

4.1 WEB SERVER RUNNING ON ESP8266

In mid of 2014 a low-cost Wi-Fi enabled chip was designed by Espressif. A simple web server is written in LUA scripting language and is run on the IC ESP8266. Below is the Figure 8 showing the web server and the module along with the server running on it.

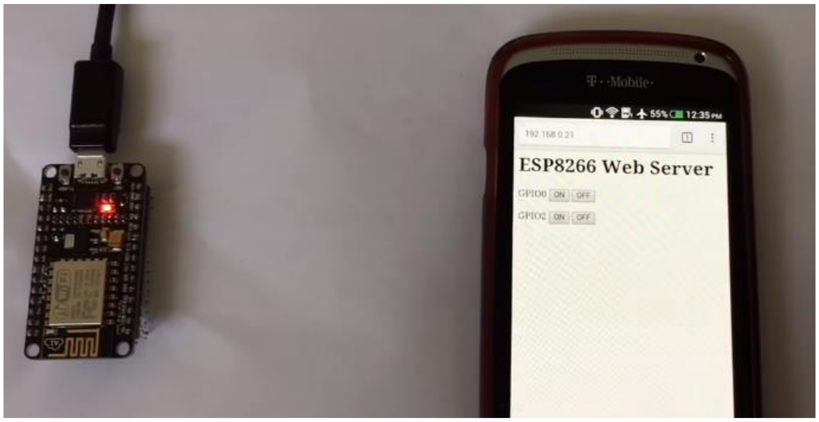
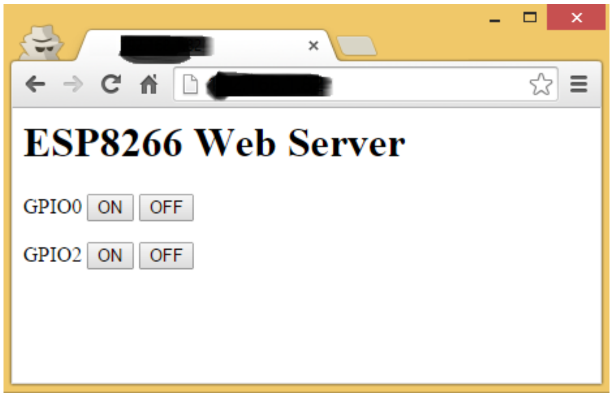


Figure 8: Testing ESP8266 by running a web server on it.

The red colored LED is turned on after the button ON is clicked on the web page responded by the ESP8266. ESP8266 is compatible with all encryption chips available on the market. This chip is very low cost and meets the minimum requirements in terms of security for the local devices or nodes which usually are located indoors and are connected to a local network.

In Fig 9 we can observe that we have control over two general purpose input output [GPIO] pins which are controlled through the web server's response after refreshing the page. The best example is the Google Nest thermostat which can be controlled and accessed by Wi-Fi. It controls the GPIO pins which in turn control the circuits for HVAC systems. Google Nest [36] houses Texas Instruments processor for further intelligence of sensing humans around.



[redacted]

[redacted]

ESP8266 Web Server

Figure 9: Web server running on ESP8266 can be accessed anywhere on the network.

The controller used here is Node MCU and it runs a very minimal version of Linux in it with LUA as its primary scripting language. It has multiple GPIO pins and supports I2C and SPI as well. In this experiment a simple LED is set to toggle with the response that is sent from the webserver. Below is the pin out of the controller used. The script for webserver is written in LUA [37] scripting language and when launched, it starts a local webserver within the network to which it is connected.

There were certain security issues which were observed during this simple experiment, such as anyone within the local network could access the server by knowing its IP address and host name. DOS attack could be easily implemented within the network and stop other users from accessing the server.

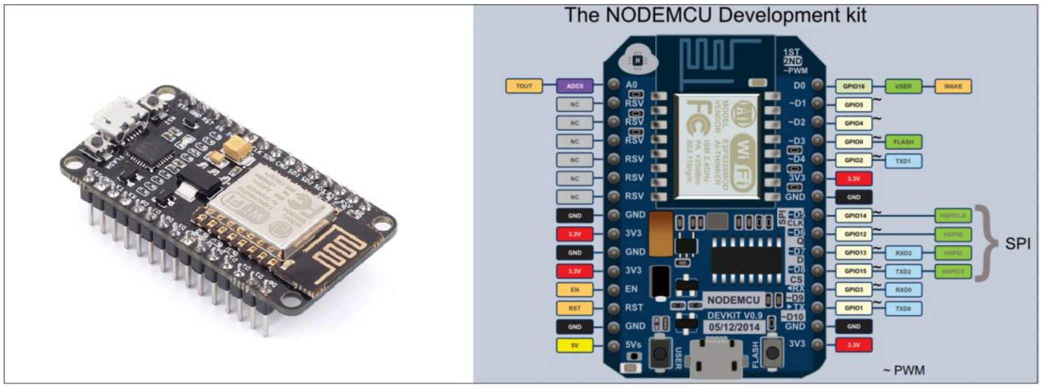
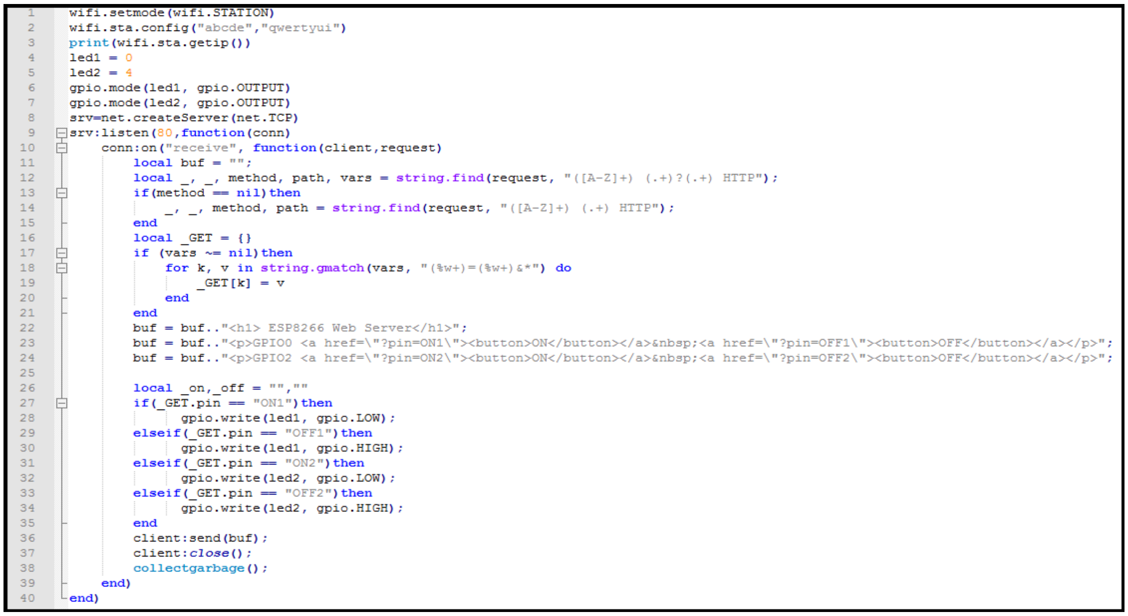


Figure 10: Pin Out diagram for ESP8266 Development Board

The script written in LUA is very straight forward without many security considerations. It fetches an HTTP request and sends a response which is visible physically via LED available on board. The wireless network's SSID and password are pre-fed into the script or a separate configuration file can hold multiple usernames and passwords such that depending on the strength of network it connects to the bets available. The two grayed out lines is the HTML part which is responsible for the visual web page.



```lua
wifi.setmode(wifi.STATION)
wifi.sta.config("abcde","qwertyui")
print(wifi.sta.getip())
led1 = 0
led2 = 4
gpio.mode(led1, gpio.OUTPUT)
gpio.mode(led2, gpio.OUTPUT)
srv=net.createServer(net.TCP)
srv:listen(80,function(conn)
    conn:on("receive", function(client,request)
        local buf = "";
        local _, _, method, path, vars = string.find(request, "([A-Z]+) (.+)?(.+) HTTP");
        if(method == nil)then
            _, _, method, path = string.find(request, "([A-Z]+) (.+) HTTP");
        end
        local _GET = {}
        if (vars ~= nil)then
            for k, v in string.gmatch(vars, "(%w+)=(%w+)&*") do
                _GET[k] = v
            end
        end
        buf = buf.."<h1> ESP8266 Web Server</h1>";
        buf = buf.."<p>GPIO0 <a href=\"?pin=ON1\"><button>ON</button></a>&nbsp;<a href=\"?pin=OFF1\"><button>OFF</button></a></p>";
        buf = buf.."<p>GPIO2 <a href=\"?pin=ON2\"><button>ON</button></a>&nbsp;<a href=\"?pin=OFF2\"><button>OFF</button></a></p>";

        local _on,_off = "",""
        if(_GET.pin == "ON1")then
              gpio.write(led1, gpio.LOW);
        elseif(_GET.pin == "OFF1")then
              gpio.write(led1, gpio.HIGH);
        elseif(_GET.pin == "ON2")then
              gpio.write(led2, gpio.LOW);
        elseif(_GET.pin == "OFF2")then
              gpio.write(led2, gpio.HIGH);
        end
        client:send(buf);
        client:close();
        collectgarbage();
    end)
end)
```

Figure 11: LUA Script to run a simple Web Server within local network

4.2 TEMPERATURE AND HUMIDITY LOGGER, REST API BASED COMMUNICATION WITH MQTT

This experiment consists of multiple hardware elements namely, DHT11 sensor which has an integrated humidity and temperature sensor under single housing and communicates via I2C protocol. The main system/development board used here is the Raspberry-Pi which runs its own flavor of Linux knows as Raspbian OS. The goal of this experiment is to understand the limits of a minimal IoT setup and what all it can accommodate without costing much. A raspberry pi is a standalone single board Linux based computer with decent processing power and memory. Here the DHT11 [38] was connected to one of the GPIO (General Purpose Input Output Pins) and power supply as well.

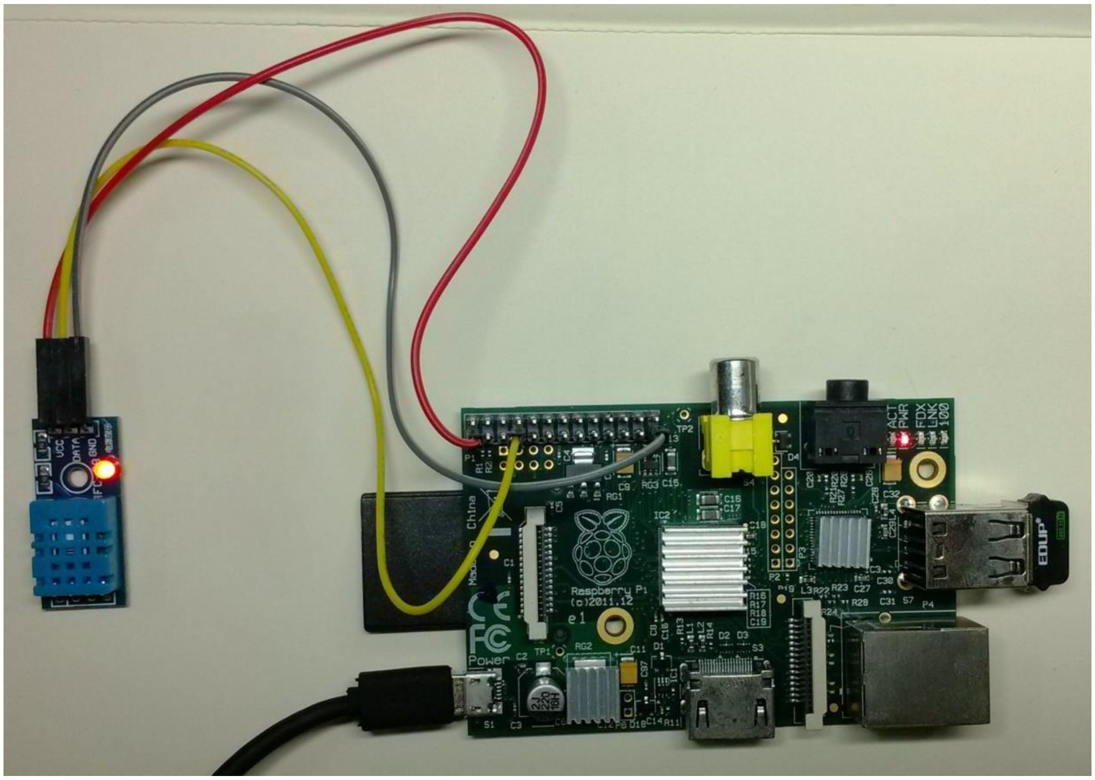
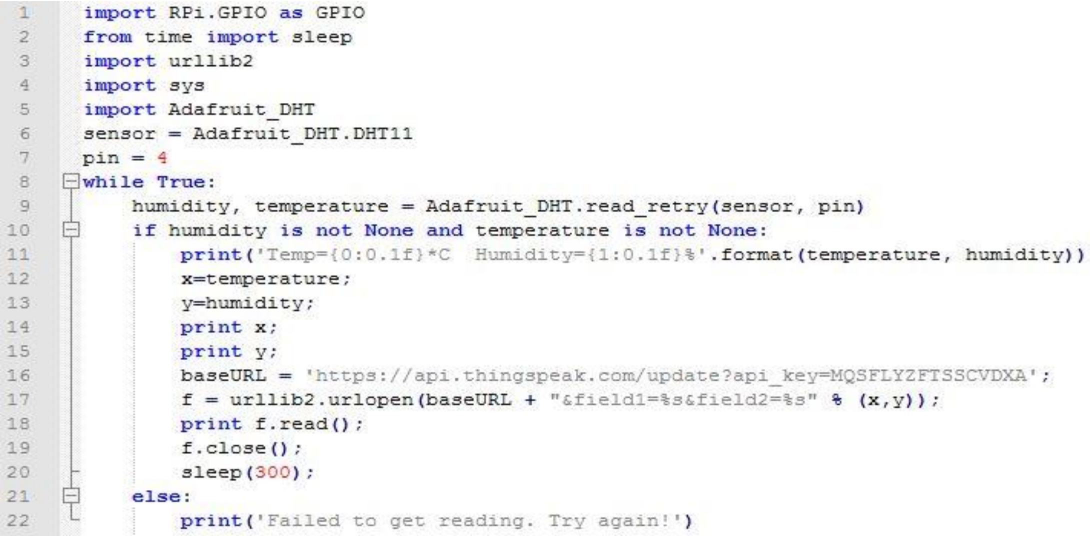


Figure 12: Raspberry-Pi with a DHT11 connected over I2C

Above figure shows three wires namely, signal, Vcc and ground interconnected between the sensor and the controller. Python scripting was used to interact with the temperature sensor. A REST API based web server was chosen to implement data push and pull operations. In Figure 12, shows the

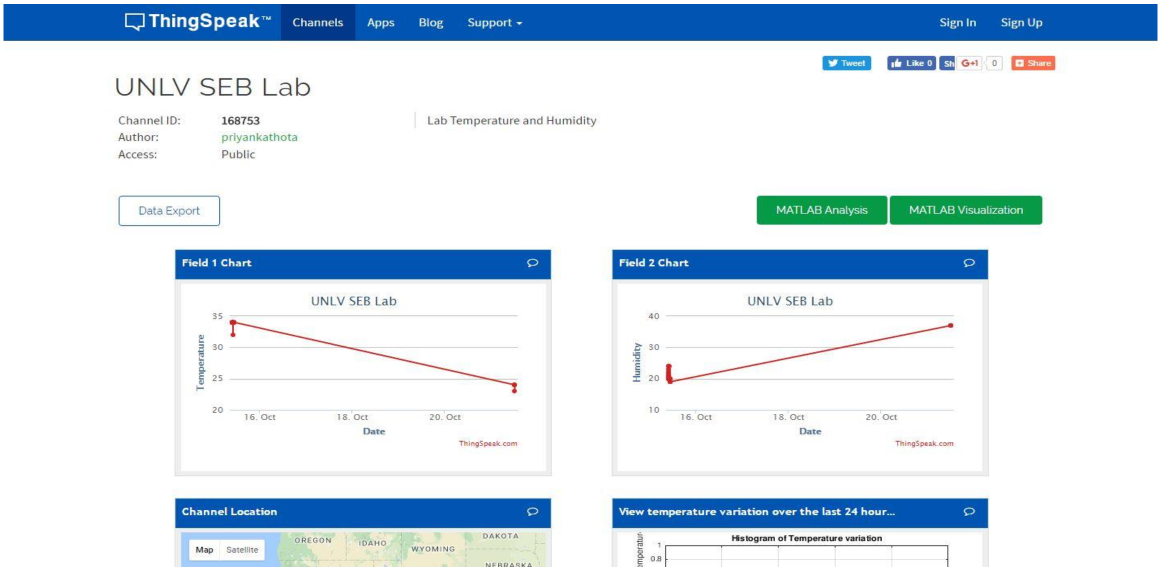
DHT11 humidity and temperature sensor connected to GPIO (General Purpose Input Output) pins of Raspberry-Pi. The python application is added to the Crontab as a new job and scheduled which repeats every 15 minutes. This calls and executes the python script continuously and sending the fresh temperature and humidity data to the server. Below Figure 13 shows the python script which contains Base URL, which is the connection string for the server.



```
import RPi.GPIO as GPIO
from time import sleep
import urllib2
import sys
import Adafruit_DHT
sensor = Adafruit_DHT.DHT11
pin = 4
while True:
    humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
    if humidity is not None and temperature is not None:
        print('Temp={0:0.1f}*C  Humidity={1:0.1f}%'.format(temperature, humidity))
        x=temperature;
        y=humidity;
        print x;
        print y;
        baseURL = 'https://api.thingspeak.com/update?api_key=MQSFLYZFTSSCVDXA';
        f = urllib2.urlopen(baseURL + "&field1=%s&field2=%s" % (x,y));
        print f.read();
        f.close();
        sleep(300);
    else:
        print('Failed to get reading. Try again!')
```

Figure 13: Python script to push data to Thingspeak

After reading the values from the sensors, they are pushed using the base URL and a write key since the default state of any channel created on Thingspeak is private. The write Key is seen at the end of base URL and this key is unique to this channel. Anyone who has this key can essentially push data into any of the fields available within the channel. Below is Figure 14 showing the graph outputs on the thingspeak website. The perks of having an API that is open sourced is, it could be run anywhere.



UNLV SEB Lab

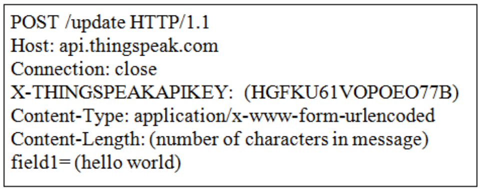
Channel ID: 168753
Author: priyankathota
Access: Public

Lab Temperature and Humidity

Figure 14: Thingspeak data visualization

4.2.1 Uploading Data

All the channels on thingspeak [39] can store data in both numeric and alphanumeric formats in up to eight different fields, and also the location details in four additional fields. Date, timestamp and unique identifier are stored with each entry. Comma-Separated Values or knows as CSV file is a very common format to store separated tuple data. Already existing data can be imported from CSV. Experimental data can be pushed into our channel immediately if existing data is not present. This can be done in HTTP by using the POST method along with a specific key to writing into a channel. In the view for channels, proceed with channels tab. From here, executing a basic program can be already achieved with the used API, as shown below.



```
POST /update HTTP/1.1
Host: api.thingspeak.com
Connection: close
X-THINGSPEAKAPIKEY:  (HGFKU61VOPOEO77B)
Content-Type: application/x-www-form-urlencoded
Content-Length: (number of characters in message)
field1= (hello world)
```

Figure 15: Code Snippet for Thingspeak API

4.2.2 Data Retrieval/Pull

Thingspeak hosted data can be retrieved by specifying a time frame or the required data ID. HTTP GET method is the most direct way to use for data retrieval. A thingspeak channel is private until unless specified public and a data read key will be required for data retrieval, which will be available in the channel view tab. The latest entry can also be retrieved by changing the given URL below. The data in the brackets would be the user specific id and the required format.

http://api.thingspeak.com/channels/(channel_id)/field/(field_id)/last.(format)

For integration into third party applications, the channel supports certain formats like .json, .xml, and .csv formats. Matlab visualizations can be added for all the 8 fields in a channel in Thingspeak. This is a very useful feature which removes the need to have a different platform to run the visualizations alone. The advantage of Thingspeak over other API's is remote accessibility of channels.

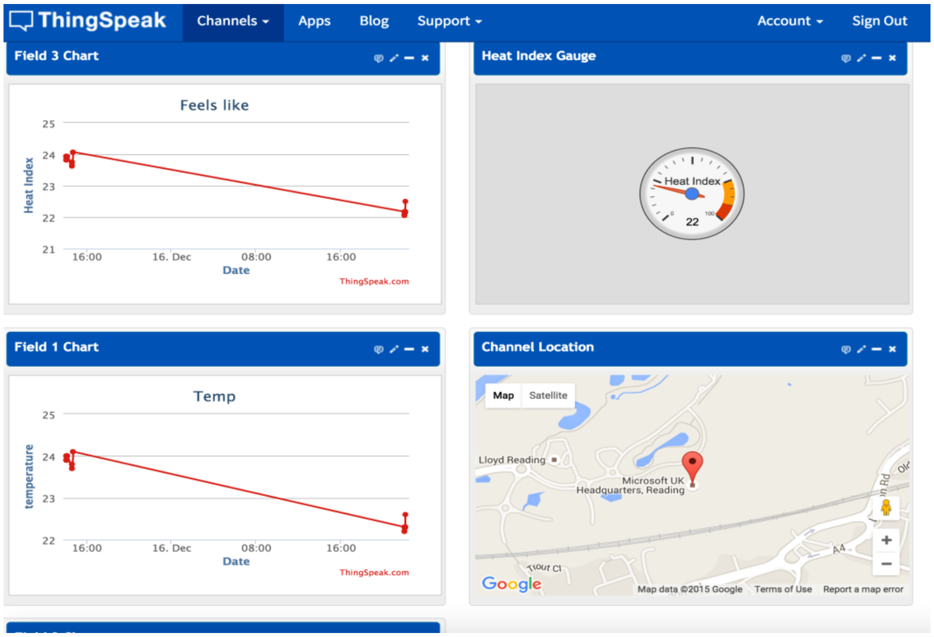


Figure 16: Dynamic representations add-ins for Thingspeak

If a channel is set to be public, then all of it fields are available anywhere just if the channel ID is known. Also, publicly accessible channels provide links for the users to download or retrieve the data in CSV, JSON and XML formats. Visualization can be as simple as a gauge meter which shows data at the moment in a quick way. Figure 6 shows the Gauge representation for temperature values which dynamically updates itself. Also, an additional field which shows location is available. The location data is pre-fed during the channel creation process.

Thingspeak is an open-source platform and hence has performance limitations due to cost constraints. It limits data upload per channel once for every 15secs. This is to avoid bandwidth consumption and traffic regulation into the server. Paid IoT interfaces [40] can handle data more frequently and in a better way. But among the open source IoT interfaces, Thingspeak performs the best.

4.3 THINGSPEAK API

Thingspeak API provides robust communication abilities to the nodes within an IoT system. It allows users to build and develop applications on the same cloud where data is stored and also provides various third party plugins which help the whole process get easy. The way Thingspeak stores data is through “channels”, each channel has 8 fields of data which are independent and there are also specific fields for showing the locational data which include parameters like latitude, longitude and elevation. This data is useful in depicting the location of the data sources for remaining 8 fields. Below Figure 2 shows the working mechanism of Thingspeak and its role in the IoT network. All the data handled by Thingspeak is date and time stamped for proper data mapping with relation to time and every data that goes in and out is assigned with a sequential ID which enables tracking data/packet loss. A ‘read key’ and a ‘write key’ are incorporated by default for every channel created and the channel is set to private.

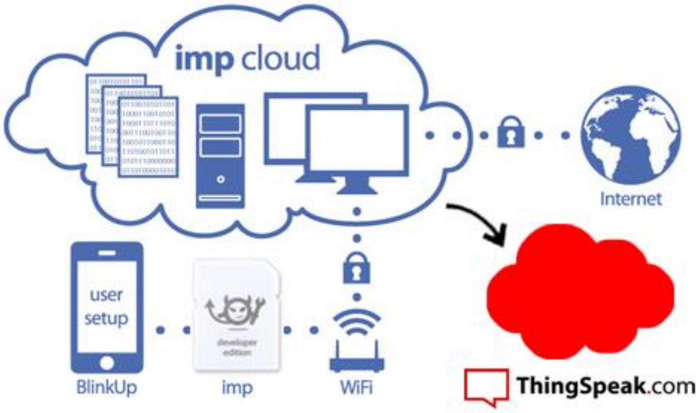


Figure 17: Thingspeak role in IoT network

The channels which are public need no read/write key to access data. According to the Thingspeak API every 'thing' is an object with sensors collecting information. Data is transferred to and fro just like how a web page would send a request using REST methods. The data is communicated here via formats like plain text, .json, .csv and .xml [41] and is pushed to any service/cloud. Once it reaches cloud, it can be accessed and used for different applications. All the collected information can be compared against a given threshold and actions or triggers can be activated if any particular field of data exceeds the threshold value. This gives a robust control for unmonitored controls like a thermostat or a high-speed fan.

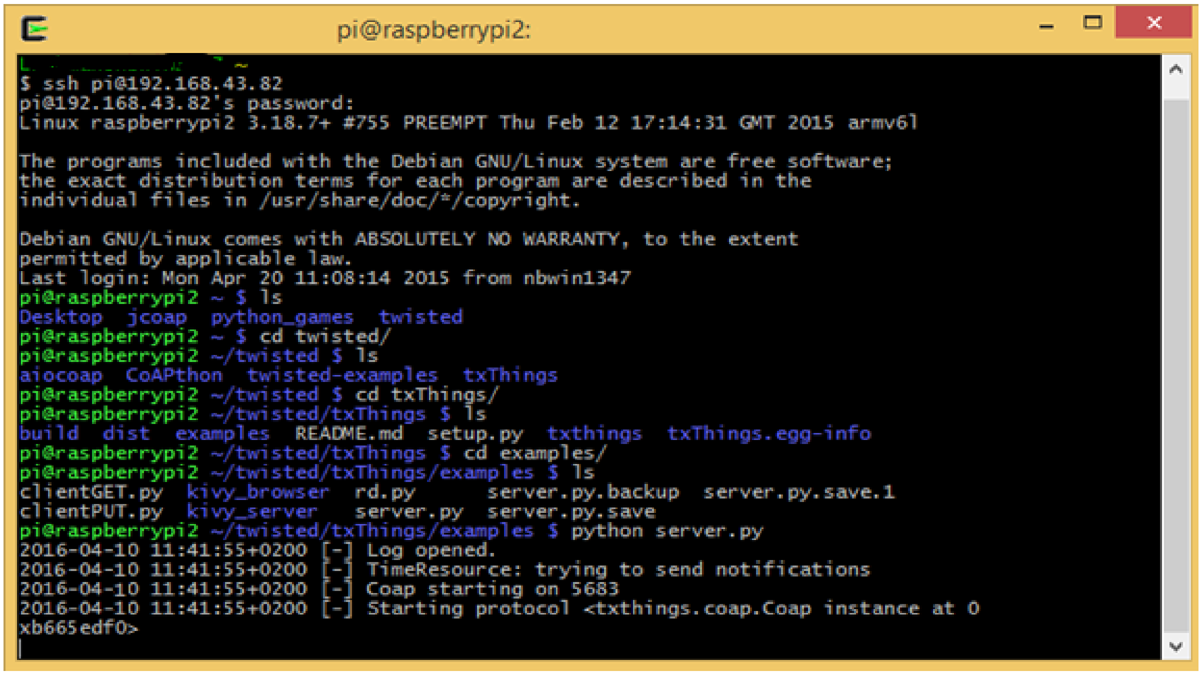
Since Thingspeak can handle 8 fields of data at a time per channel, visualization becomes easy and a graph representing all 8 parameters can convey lot more than a graph with single parameter. There are multiple reasons for why the Thingspeak API for IoT was selected for analysis of this research among other available APIs like 'sensorthings', 'skynet' and 'smartobject'. Compared to other API's Thingspeak is more appealing as it is open source and has much better data visualization using spline graphs. It uses an application server named Phusion Passenger Enterprise. Hence it provides good support for other languages as well which are popular into IoT, example: ruby, python and node.js.

4.4 SECURITY ISSUES

In this experiment the security issues were minimal since a key value was used for writing or reading the data from the server. Apart from the keys, there is also a channel ID which is unique for every channel created. Hence for one to access the data or write new data there is a two-factor authentication required, until unless the channel is set to public, in the case where data is accessible to everyone without authentication. DOS attack is also less likely to happen on the server side since one could inject data only if the channel ID and Authentication Read/Write keys are known [42].

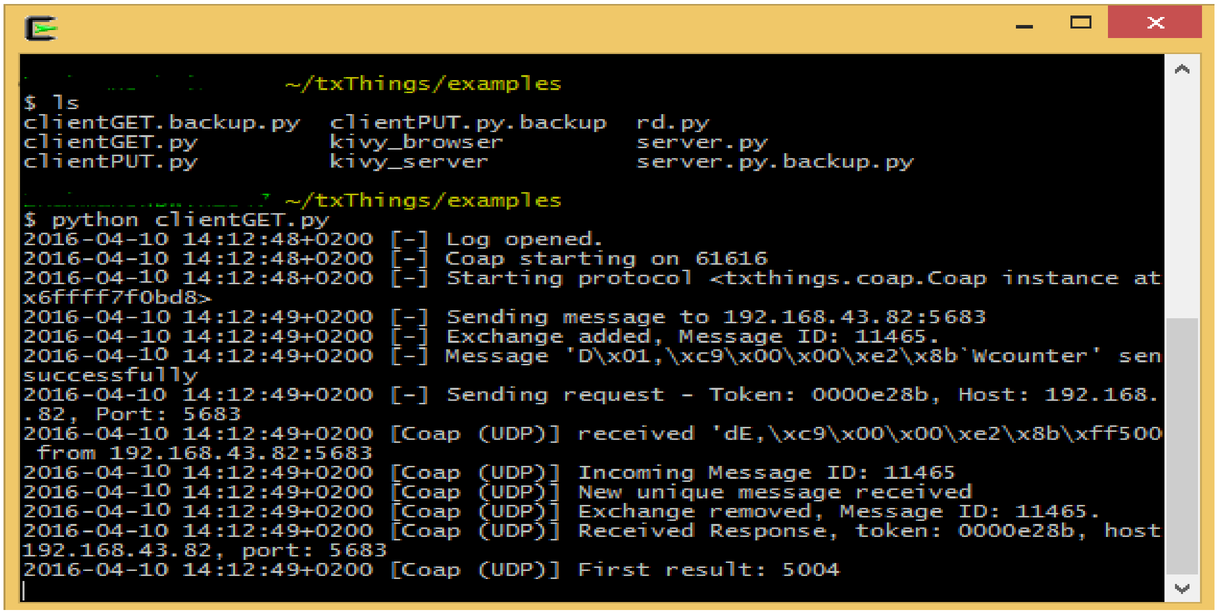
4.5 MINIMAL CoAP SERVER

A simple implementation of the CoAP protocol can be done on Raspberry-pi. The framework used for CoAP is "txThings", which is a python implementation of CoAP. The library contains three examples server.py, client_GET.py and client_PUT.py.



```
$ ssh pi@192.168.43.82
pi@192.168.43.82's password:
Linux raspberrypi2 3.18.7+ #755 PREEMPT Thu Feb 12 17:14:31 GMT 2015 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Apr 20 11:08:14 2015 from nbwin1347
pi@raspberrypi2 ~ $ ls
Desktop  jcoap  python_games  twisted
pi@raspberrypi2 ~ $ cd twisted/
pi@raspberrypi2 ~/twisted $ ls
aiocoap  CoAPthon  twisted-examples  txThings
pi@raspberrypi2 ~/twisted $ cd txThings/
pi@raspberrypi2 ~/twisted/txThings $ ls
build  dist  examples  README.md  setup.py  txthings  txThings.egg-info
pi@raspberrypi2 ~/twisted/txThings $ cd examples/
pi@raspberrypi2 ~/twisted/txThings/examples $ ls
clientGET.py  kivy_browser  rd.py      server.py.backup  server.py.save.1
clientPUT.py  kivy_server   server.py  server.py.save
pi@raspberrypi2 ~/twisted/txThings/examples $ python server.py
2016-04-10 11:41:55+0200 [-] Log opened.
2016-04-10 11:41:55+0200 [-] TimeResource: trying to send notifications
2016-04-10 11:41:55+0200 [-] Coap starting on 5683
2016-04-10 11:41:55+0200 [-] Starting protocol <txthings.coap.Coap instance at 0
xb665edf0>
```

Figure 18: CoAP Server running on Raspberry-pi.



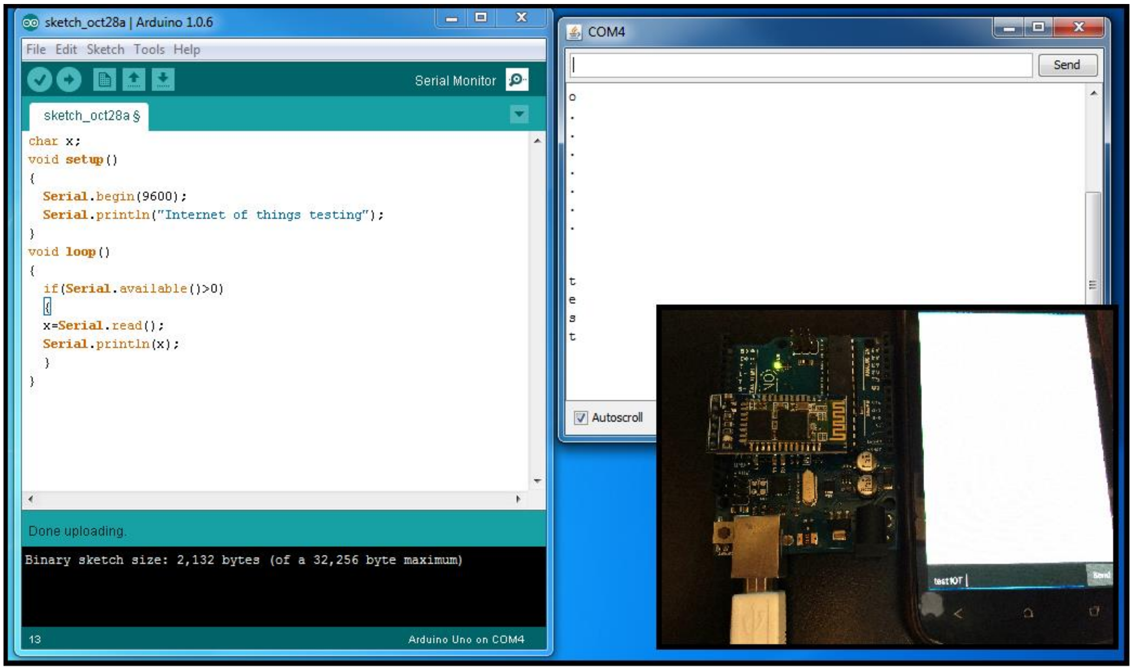
```
[redacted]          ~/txThings/examples
$ ls
clientGET.backup.py  clientPUT.py.backup  rd.py
clientGET.py         kivy_browser         server.py
clientPUT.py         kivy_server          server.py.backup.py

[redacted] ~/txThings/examples
$ python clientGET.py
2016-04-10 14:12:48+0200 [-] Log opened.
2016-04-10 14:12:48+0200 [-] Coap starting on 61616
2016-04-10 14:12:48+0200 [-] Starting protocol <txthings.coap.Coap instance at
x6ffff7f0bd8>
2016-04-10 14:12:49+0200 [-] Sending message to 192.168.43.82:5683
2016-04-10 14:12:49+0200 [-] Exchange added, Message ID: 11465.
2016-04-10 14:12:49+0200 [-] Message 'D\x01,\xc9\x00\x00\xe2\x8b`Wcounter' sen
successfully
2016-04-10 14:12:49+0200 [-] Sending request - Token: 0000e28b, Host: 192.168.
.82, Port: 5683
2016-04-10 14:12:49+0200 [Coap (UDP)] received 'dE,\xc9\x00\x00\xe2\x8b\xff500
 from 192.168.43.82:5683
2016-04-10 14:12:49+0200 [Coap (UDP)] Incoming Message ID: 11465
2016-04-10 14:12:49+0200 [Coap (UDP)] New unique message received
2016-04-10 14:12:49+0200 [Coap (UDP)] Exchange removed, Message ID: 11465.
2016-04-10 14:12:49+0200 [Coap (UDP)] Received Response, token: 0000e28b, host
192.168.43.82, port: 5683
2016-04-10 14:12:49+0200 [Coap (UDP)] First result: 5004
```

Figure 19: CoAP client running on a remote computer.

The first program would start a CoAP server on local host and the example client performs a GET request to local host and the last file performs PUT request to local host. The txThings framework is installed on Raspberry-Pi. The server is run on Raspberry-pi and the client is run on a desktop using Cygwin for windows. Figure 18 and figure 19 show the CoAP server and the client running.

4.6 COMMUNICATION VIA BLUETOOTH USING HC-04 AND ARDUINO

A HC-04 Bluetooth module was used for sending texts and data through an Android smart phone. This test analyzed the data and the encryption applied on it. Optional pre-shared key authentication and encryption algorithms are used by Bluetooth links which are widely considered strong when both used and implemented correctly. Length and randomness of the passkey used for Bluetooth pairing determines the strength of Bluetooth security, during which the devices mutually authenticate each other for the first time. They also set up a link key for the authentication and encryption later. Discoverability and connect ability settings are also important for Bluetooth [43] security. The user authorization for incoming connections provides additional security.



```
char x;
void setup()
{
  Serial.begin(9600);
  Serial.println("Internet of things testing");
}
void loop()
{
  if(Serial.available()>0)
  {
  x=Serial.read();
  Serial.println(x);
  }
}
```

```
Done uploading.
Binary sketch size: 2,132 bytes (of a 32,256 byte maximum)
```

```
13                                         Arduino Uno on COM4
```

```
o
.
.
.
.
.
.
.
t
e
s
t
```

Figure 20: Arduino and HC-04 interaction via Serial Port

In Figure 20 it shows the successful communication between terminal applications running on android device. Data is tested through two nodes using Bluetooth serial communication. The distance is limited and the baud rate cannot go beyond 11200, which is very low for some application where higher bandwidth is required. Only one-to-one communication could be made and if another node sends a request to the application running on the android platform, it would simply reject or ignore the second node. There exists a "Link Layer" in the Bluetooth communication protocol. This link layer has four different entities used for maintaining security.

1. A Bluetooth device address, BD_ADDR (48 bits)
2. A secret authentication key (128 bits)
3. A secret encryption key (8-128 bits), the encryption key is derived from the authentication key. Each time encryption is activated; a new key shall be generated.
4. A pseudo-random number, RAND (128 bits), it will be generated for each new transaction. Each device has a pseudo-random number generator to generate pseudo-random numbers.

The Link layer is transparent to the security controls imposed by the application layers. Within the Bluetooth security framework, user-based authentication and fine-grained access control can be enforced. The packets are checked for wrong delivery or errors by using the channel access code and the CRC [44] in the payload. A packet may consist of:

1. Access code
2. Header
3. Payload

CHAPTER 5

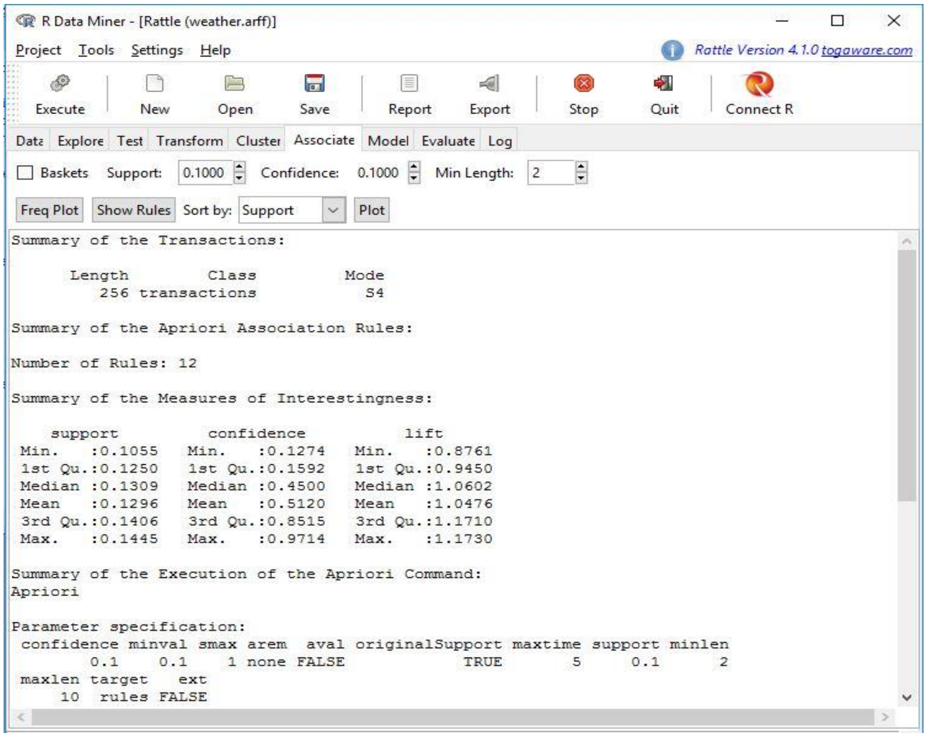
PREDICTION AND DATA ANALYSIS USING RATTLE

5.1 PREDICTION ANALYSIS

Rattle is a GUI library for R, which itself is a powerful platform for data mining. R is mainly used for challenging tasks that are related to pattern identification and mining of data. R gives people the same level of computing power which other commercial analytic software's would give. It is a language like any other. It is mainly for the highly skilled statisticians. Rattle uses tab based concepts. The standard flow for a data analysis related project can be:

1. Input large chunks of data and required parameters
2. Understand the data and its distributions
3. Verify distributions
4. Transform the data
5. Develop some models
6. Check models and datasets
7. Verify event log for the whole process

For understanding Rattle [45] better, we have taken a sample IoT data related to weather. This includes data from temperature, humidity and pressure sensors. Using sample data that could have been collected for a long period of time can give us good analysis of how the data flow behaves and the frequency of values or data which may repeat. This makes easy identifying patterns or predicting the upcoming set of data. Essentially a behavioral model can be conceived using existing data. Different association rules can be applied which give the support and confidence values for a given data set and the required outcome. Data can be imported into the R console in multiple ways and formats. Rattle provides support for comma separated values, plain text, .arff (This is a common format), and ODBC connections. The data attached to the current session and all the datasets which are available to use from the packages are installed in the libraries section and are provided along with the interface itself.



□ Baskets Support: 0.1000 Confidence: 0.1000 Min Length: 2

Sort by: Support

```
Summary of the Transactions:

      Length         Class          Mode
         256 transactions            S4

Summary of the Apriori Association Rules:

Number of Rules: 12

Summary of the Measures of Interestingness:

    support           confidence           lift
 Min.   :0.1055   Min.   :0.1274   Min.   :0.8761
 1st Qu.:0.1250   1st Qu.:0.1592   1st Qu.:0.9450
 Median :0.1309   Median :0.4500   Median :1.0602
 Mean   :0.1296   Mean   :0.5120   Mean   :1.0476
 3rd Qu.:0.1406   3rd Qu.:0.8515   3rd Qu.:1.1710
 Max.   :0.1445   Max.   :0.9714   Max.   :1.1730

Summary of the Execution of the Apriori Command:
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
        0.1    0.1    1 none FALSE            TRUE       5     0.1      2
 maxlen target   ext
     10  rules FALSE
```

Figure 21: Support and confidence for a given dataset

Figure 21 shows the support and confidence values for a sample weather data. Support is an indication of how frequent a given value or a variable is in a given database/dataset. Higher support indicates higher chances of hit for a given value. The support value of any object with respect to other objects in a database shows the presence of former object in every transaction of the later object, for all entries in the database.

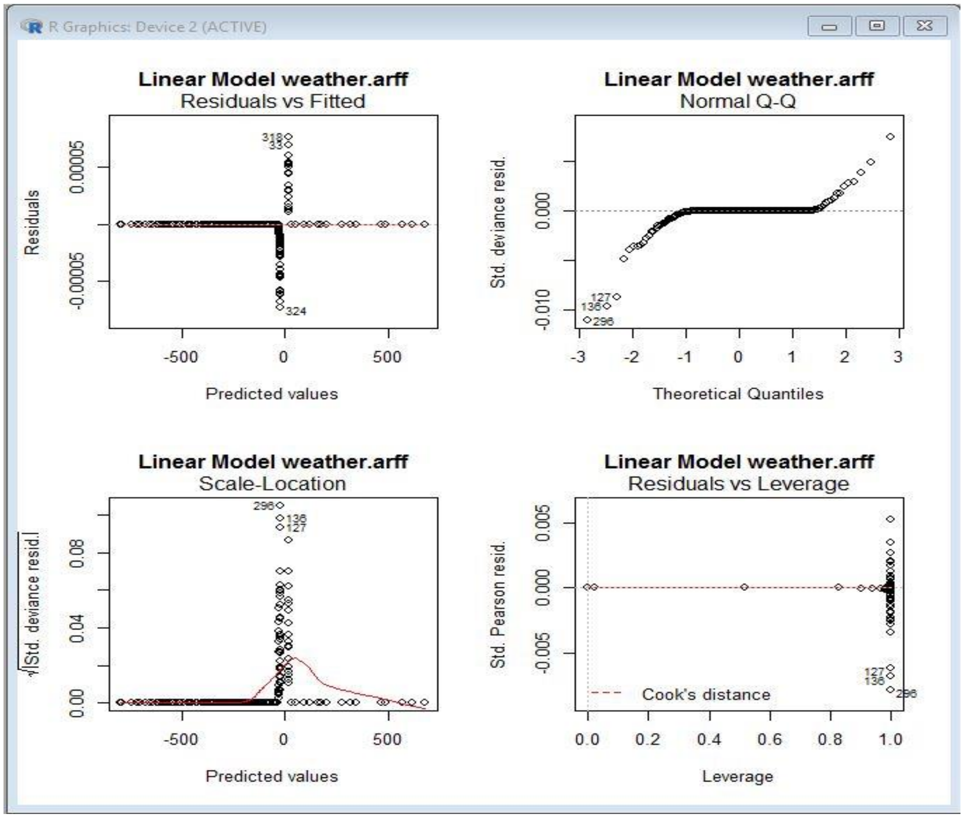


Figure 22: Predicted Values and Leverage plot

Further R features various representations for large datasets using graph plots with various algorithms. Figure 22 is the outcome of our sample IoT data which shows prediction values and leverage values. On the other hand, Confidence accounts to how many times a given value that showed up was true. There are algorithms to evaluate these values on a given dataset like Apriori, Elcat [46] and FP-growth. Rules for association need to meet the requirements of a user support and minimum confidence at a time. The rule creating process is divided into two tasks:

1. A threshold value is used to identify all frequent data items in a given collection
2. A fixed confidence value is required to make a rule based on dataset.

5.2 DATA ANALYSIS

For most of the part an IoT system may seem a simple node to node communication system or a part of a bigger system. What comes along with IoT is data, which is huge in size and can be very useful. In this research, every aspect of IoT has been covered along with experiments, analysis and conclusions. Along with those, data analysis also has been performed on data sets which were collected during experiments. This data is very important since it reveals the hidden patterns within the data. The best way to analyze it could be considering a system which deploys IoT in large scale, for example, Home Automation systems. These systems are vast and spread across thousands of homes involving many sensors for each home like temperature sensors, microphones, and appliance control equipment.

Over a period of year or few months one can collect data of all these units periodically and sample them based on the precision of data required. This data when visualized using tools like Rattle, can give very useful metrics which can be used to modify the system to increase its efficiency. In a given city of 1000 people, IoT data can help analyze the temperature patterns of people during given climates at particular locations. This can help divert more power to cities where consumption is higher and adequate power to rural areas where the requirement is less. Data analysis can save lots of costs for the system maintenance and improve the way entire system works.

CHAPTER 6

COMMUNICATION MODES, PARAMETERS AND THEIR LIMITATIONS

6.1 COMMUNICATION MODES

Major modes of communication for IoT or any internet connected system with multiple nodes are:

1. Satellite
2. Wi-Fi
3. Radio Frequency
4. Radio Frequency Identification [RFID]
5. NFC [47]
6. Bluetooth

These protocols allow manufacturers to select the mode of communication in a wide aspect considering the requirements and cost factors. Cost effective data nodes will help deploy the IoT network faster and keep the maintenance costs limited. An IoT system may consist of combinations where multiple technologies and protocols can synchronize.

6.1.1 Satellite

Satellites play a very major role in the growth of IoT. If the devices are geographically separated by a long distance, satellites are the only way to establish a connection between them. In a wide range of applications across the industries and across the geographical borders, from the most basic mining to energy management sector, these can provide prominent advantages than the other ways of communication. In mining industry, they use it for their exploration projects, in the connected cars etc.

6.1.2 Wi-Fi

Wi-Fi is one way of connecting the nodes in IoT. This can be used for the devices in a particular range. The purpose served by satellites cannot be served by Wi-Fi but satellites are not always the solution as they are of high cost. Advantages of using Wi-Fi module:

1. Wi-Fi access points can be directly connected to the internet without any change in the module.
2. All the smartphones already have an embedded Wi-Fi module.
3. Compact form factors
4. Rapid connection setup times
5. Massively scalable deployments
6. IP-based communications
7. Security and Integrity
8. Flexibility

6.1.3 Radio Frequency (RF)

In some applications, Wi-Fi is too complicated to be used in the sensor technology. In such cases, Radio Frequency is used as an option to connect the nodes in the IoT. Advantages of using Radio Frequency:

1. Low cost
2. Low power consumption
3. High transmission range
4. Better indoor performance

6.1.4 Radio Frequency Identification (RFID)

The devices to be connected by Internet of things (IoT) should have an Auto-ID technology which is called as the RFID tag. This ID makes sure that the object is uniquely identified. With this technology the data can be monitored. The devices connected through RFID in IoT send data to a central server which stores the data in human-readable format.

6.1.5 Near Field Communication (NFC)

It provides high levels of convenience and interaction among all the devices in the home. All the devices can be added to a home network and can be controlled with our mobile phones and tablets.

NFC gives a tap-and-go experience. NFC devices can register themselves on their own. They can remember their warranty dates. They can remind people of when a service is required. Advantages of NFC technology:

1. Personalized settings
2. One-step payments
3. Efficient data tracking
4. Easy access to online maintenance

6.1.6 Bluetooth

The presence of Bluetooth in our smartphones makes it easy for the developers to design a service based on it. The range is a limitation for Bluetooth. It is so energy and power efficient. It is perfect for the devices that should run with less battery for long periods of time. For Example, we can go for a run after waking up. Then, a heart rate monitor can directly speak to our smart watch and keep control of things. We can brush our teeth with a sensor enabled toothbrush while listening to music through the showerhead.

Among all the above mentioned modes of communication, satellites are used for large networks and remaining is used for modular communication. The weakest mode of communication which can be hacked or attacked by intruders is Wi-Fi and weakest mode to jam is RF. Every mode of communication mentioned above has its own drawbacks in the area of security. Recently, NEST learning thermostat has been developed which is an energy conserving device that optimizes temperature control at homes and industries. It is a self-learning Wi-Fi enabled thermostat [48]. This device is connected to a home network and poses very little potential threats. Even if it is hacked or taken into control, the intruder may have the control of temperature at the most.

6.2 COMMUNICATION PARAMETERS

6.2.1 Baud Rate/Data Speed

This is an important factor if the number of tuples of data sent by the sensor or a node is high. For example, if a temperature value is the only thing to be sent, it can be done with extremely low bandwidth network. If some data generated by a 9-Axis Gyroscope, Accelerometer has to be sent 100 times a second to maintain precision; in such cases higher bandwidth is required.

6.2.2 Equipment/Hardware Durability

The sensor or the node itself has to be rigid and should not expose any of its electrical contacts to the outside world. Since this will allow intruders to bypass the inbuilt layers of security and directly corrupt data/system over the wire

6.2.3 Hardware Cost

When a system has to be designed which consists of thousands of nodes for data collection, the cost per node for implementation is very critical since it is common for some of the nodes to be down or get damaged and if the cost per node is very high then replacement of nodes will be very difficult and maintenance of the entire system becomes unfeasible.

6.2.4 Power Requirement

This is another crucial factor in considering a situation where hundreds of nodes are spread across vast area, then connectivity for communication can be done via a feasible wireless technology but power supply can't escape the wires. Hence if a node is capable of working well with low power consumption which can be provided by a solar panel let us say. In such cases it would be really convenient to power each node from where it is located. If a node is power hungry then again it increases overall system cost in terms of power consumption as well as cabling and maintaining power cables.

6.2.5 Distance/Range or communication

This comes under one of the important factors to be considered before setting up an IoT system. Depending on the environment example farming may require its humidity sensors to be spread across hundreds of acres of area. In such situation a suitable communication mode has to be chosen which essentially is wireless and also offers a long range communication so that amplifiers and repeater costs can be avoided.

6.3 LIMITATIONS AND ADVANTAGES

Every technology mentioned above has its own limitations and use cases. Wireless tech is limited by its range and strength, where else wired tech is limited by its power, installation costs and wear and tear. As mentioned above there are different parameters which have to be considered before selecting any protocol/technology for a given IoT system. Certain systems may be designed with consumer convenience as priority, for example NFC is used in smartphones for payments and other similar transactions. One may question why not use Wi-Fi or Bluetooth which is already installed on a smartphone. Using an available mode like Wi-Fi for payment may require the user to get connected to the internet during the payment and also will need some implementation on the security part, since the transaction will travel back and forth the server and needs to be secure. This will need a dedicated complex application with all features in order to keep the application secure and reliable. Doing such will increase the total cost of maintaining the application and making it limited with different constraints.

Hence NFC would be a good choice in such scenario where authentication is inbuilt and valid internet connection is not a requirement, since transactions can be queued up and come to effect when the internet connection is back. Cases like this will need a broader understanding of using an efficient protocol and using an effective mode of communication.

Large IoT systems may use multiple protocols and modes to achieve the overall optimal throughput. New modes of communication between modules have been evolving and hardware which is compatible with these is also being researched.

Every mode of communication comes with its own limitations, after conducting experiments certain limitations were evident, which are listed below.

6.3.1 Bluetooth

1. Distance

Most of the Bluetooth modules are found inside portable devices like smartphones and laptops, and these devices are powered by battery. The modules used in these systems belong to Class-2 Bluetooth devices which have a range of 10 meters or 30 feet. Class-1 Bluetooth devices are more power hungry and bulky which can communicate up to 100 feet but are not feasible to be used in small portable devices.

2. Slow Data Transfer Rate

The rate of data transfer in terms of bits between most common Bluetooth devices is around 3 Mbps. This is significantly lower than Wi-Fi and other protocols [49]. There are different versions of Bluetooth and each of them has different speeds. Latest versions use Bluetooth in a combination with Wi-Fi in order to stream music or files across two devices.

3. Interference

There is a frequency at which these radios work and Bluetooth devices operate at 2.4GHz which also is used by many other wireless devices. Bluetooth is designed to hop frequencies for data transfer multiple times a second in order to avoid interference, but if multiple devices are trying to use the same bandwidth with similar principles, it becomes difficult to avoid interference. This, in turn, will make the data transfer slow as files or requests have to be sent multiple times. Modern Bluetooth 4.0 etc. use different frequencies higher than common radios around 6-9 GHz.

6.3.2 Wi-Fi

Physical Objects are one of the significant things which determine the signal strength of Wi-Fi. Depending on the density of materials used, it becomes difficult for Wi-Fi to pass through certain materials. Interference is a factor for Wi-Fi as well, since it operates on 2.4GHz and 5 GHz. Devices like Microwave Ovens emit high power signals at 2.4GHz and most of it is random noise which affects the whole area enough to bring down the efficiency. A number of users and the share of bandwidth impact the overall speed which individual user is able to get. Number of peers connected to a Wi-Fi system will lead to slow speeds for all. Wi-Fi has multiple networks and a mixed network like 802.11b/g/n can affect the speed of data flow especially when people get connected from distinct networks.

6.3.3 NFC

The major limitation with NFC would be security. It is easy to clone NFC or chip ID using an NFC reader. This can be dangerous when making payments, since the NFC reader may be duplicated to collect data. NFC is new technology, not many manufacturers are comfortable with using it.

Similarly, Satellite, RFID, Radio Frequency altogether share common disadvantages of interference, power and ranges. Satellite communication will have extremely large range compared to all other modes of communication, but the speed or bandwidth at which one can communicate would be less depending on the distance. There are some rare modes of communication like laser communication which will be extremely fast but again has a downside where it requires clear line of sight. Changes in air temperature can affect the refractive index of medium which in turn will make almost impossible to communicate through laser during hot days. Wide receptors have to be deployed to receive the distorted light signals and correct them.

CHAPTER 7

SECURITY ISSUES

Largely connected networks are prone to multiple attacks and drawbacks. The common emerging hardware module for IoT is the ESP8266, which is the cheapest Wi-Fi module available which can communicate in serial with the micro-controllers and communicate over the connected Wi-Fi network. Modules similar to ESP8266 are installed on various IoT devices like temperature logger or smart health tracker. When more timers are loaded onto this chip it misses many events automatically and becomes difficult to analyze the problem as the software part is closed. Due to lack of memory it is difficult to run TCP-IP stacks and when it comes to HTTPS it's almost impossible due to small form factor and less memory. These modules are more integrated into IoT products every day and are limited in such a way that even if one HTTPS connection is made, it can't handle another one at that moment. The maximum encryption this new chip can do is about 512bit encryption [50] and nothing greater than that is possible. This module is more suitable for internal home network Wi-Fi where security concerns are limited and incase this is to be implemented over the internet, then something with real TCP-IP stack and good security like RT5350-OLinuxino single board computer can be used. This now can handle 2048-bit key and hide well your devices which otherwise are prone to attacks by the intruders.

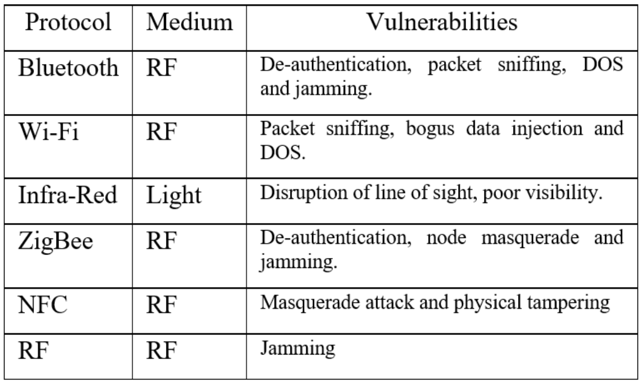
7.1 VULNERABILITIES

A network is always vulnerable to various attacks like denial of service attack, injection of bogus data/malware and packet sniffing. These attacks happen in the existing internet every day and have their own defense mechanism too. Antivirus programs prevent any malware from affecting the systems. In most of these attacks on the internet either the user's private data is collected or the required service is blocked. This is not the case in IOT, here we have data nodes which are either sensors or things equipped with a communication module. There are multiple ways for a system to be affected and it can pertain to single node or whole system.

The possible vulnerabilities are:

1. Take Control of the system
2. Steal Information/Data Packets
3. Disrupt/Stop services

Taking control can be disastrous as an intruder may hack into a self-driving car and take control and cause accidents. Stealing of information also happens the same way where the intruders can pull out valuable information such as related to the specifications of a new car can be sneaked out by hacking the sensors of a manufacturing plant. There are various ways through which security must be addressed throughout the lifecycle, from the initial design to the operational environment. Below the Table.1 shows different protocols and the possible attacks.



Protocol	Medium	Vulnerabilities
Bluetooth	RF	De-authentication, packet sniffing, DOS and jamming.
Wi-Fi	RF	Packet sniffing, bogus data injection and DOS.
Infra-Red	Light	Disruption of line of sight, poor visibility.
ZigBee	RF	De-authentication, node masquerade and jamming.
NFC	RF	Masquerade attack and physical tampering
RF	RF	Jamming

Table 4: Protocols and their vulnerabilities

7.2 DEFENSE

Below mentioned measures are preliminary things to be considered when a device is designed for IoT. In Figure 4 shows the proper design paradigm for any device in the IoT. Small systems may not follow or implement every component which is listed below, but any IoT system which has secure data transactions and deals with crucial data needs each of these components to make sure the whole system stays reliable and efficient. Single security flaw can bring down an entire system. For example, if the humidity sensor network is hacked and values are morphed, entire crop can suffer from improper water feed or excessive water, either way destroying the crops and leading to heavy loss.

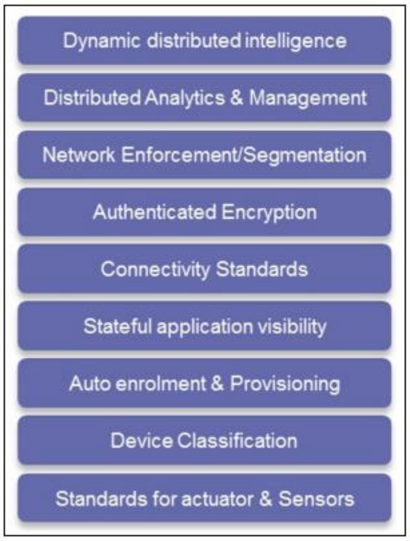

Dynamic distributed intelligence
Distributed Analytics & Management
Network Enforcement/Segmentation
Authenticated Encryption
Connectivity Standards
Stateful application visibility
Auto enrolment & Provisioning
Device Classification
Standards for actuator & Sensors

Figure 23: Proper Design principles for IoT

7.2.1 Secure booting

The digital signatures generated cryptographically are used to verify the integrity and also authenticity of the software that is on the device when the device is supplied with power. Just like how a person signs a legal document or a check, the software image needs a digital signature and it is to be device verified which ensures that the software is authorized to be run on the device. It also has to be signed by the entity which authorized it, which will be loaded. All this establishes the foundation of trust, but the device still has to be protected from several malicious intentions and run-time threats. At the time of manufacturing many modules in Internet of Things usually have their hardware firmware flashed.

1. Access control

Next, various forms of access control and resources are applied. The privileges of applications and device components are limited by the role-based access or mandatory controls that are built into the operating system so that only the resources needed to do their jobs can be accessed. Access control is used to ensure that even if any component in the system is compromised, there would be minimal access for the intruder to other parts of the system. Network-based access control systems like Microsoft Active are analogous to directory device-based access control mechanisms. To gain access to a network, even if corporate credentials are managed to be stolen, the compromised information would be limited to only the particular areas in the network that are authorized by particular credentials. According to the least privilege principle, only the minimum access needed to perform a task is to be authorized in order to minimize the effectiveness of any security breach.

2. Device authentication

A device should authenticate itself before transmitting or receiving data when it is plugged into a network. There won't often be users sitting behind keyboards of the deeply embedded devices, waiting for the credentials to be entered that are required to access the network. Then how can we ensure that the devices are correctly identified prior to the authorization? Machine authentication will allow a device to access the network according to the credentials that are stored in a secure area, just the way a user authentication allows the user to access a corporate network based on username and password.

3. Firewalling and Intrusion Prevention Systems

The device also requires a deep packet inspection or firewall capability in order to control the traffic that is to be terminated at the device. If the network-based appliances are in place why is a host-based firewall or Intrusion Prevention Systems required? Deeply embedded devices have exclusive protocols, that are very distinct from the regular enterprise IT protocols. For example, the smart energy grid has its own set of rules that govern how devices in the network communicate with each other. The device need not concern itself with filtering any common and high-level Internet traffic. The network appliances should take care of it, but it has to filter the particular data destined to terminate on that device.

4. Updates and patches

A device will start receiving hot patches and software updates once it is in operation. The patches need to be rolled out by operators, and the devices have to authenticate them, in a way that does not utilize bandwidth or destroy the functional safety of the device. Windows users receive updates from Microsoft and tie up their laptops for fifteen minutes. Thousands of devices would be performing critical services or functions and are dependent on the security patches to protect themselves against the inevitable vulnerability. Security patches and software updates must be delivered in a way that requires the intermittent connectivity and limited bandwidth of an embedded device and eliminates the possibility of compromising functional safety. Patches must be released as soon as bugs are detected.

7.3 AUTHENTICATION, ENCRYPTION AND INTEGRITY

7.3.1 Authentication

This process is a part of communication when a user or a node is trying to communicate with another user or node. This is a requirement since the authenticity or whether the intended node/person are accessing the data should be known. Authentication can be as simple as a username and password or could have a randomly generated hash key which could be verified on the other end. Authentication can be considered as a basic token of verification if a given element is the intended one or authorized one. This process will make sure that data doesn’t go into wrong hands. In case of IoT authentication [51] does not only happen between a user and a system but also happens between two different hardware IoT nodes. This is because any node despite its security and safety could get corrupted or affected by a malware that can creep into network. If authentication between nodes is not implemented, in such case there is every chance for that corrupt node to transmit the malware and affect all remaining nodes.

7.3.2 Encryption

This is the next step. For data being sent back and forth between two nodes has to be safe even if authentication fails at any point. Some of the cases, the data traveling through the system is more valuable than the system itself. In such cases encryption comes handy. Encryption happens differently on different modes of communication. Bluetooth has its own encryption standards, RF transmitters may use encoders and decoders with frequency hopping and other techniques to make data transfer complex and hard to understand for any intruder even if they get successful in entering the system. Encryption has to be implemented at every port of exit, it could be a small node or a large node. Every point which sends out data needs encryption, since a single unprotected point can be dangerous for a system.

7.3.3 Integrity

Making sure that encryption and authentication are implemented well, these things ensure the integrity of data being transferred. The integrity of data is not lost when the data being sent is untouched and unmodified during transfer [52].

CHAPTER 8

CONCLUSION

This research conducted various experiments in different environment to analyze IoT communication modes and protocols. Data analysis was performed on certain data sets that were collected through different sensors and this was used identify the changes in patterns of the collected data. This analysis gives a deeper insight into specific protocols like MQTT and CoAP which are the prominent protocols for IoT today. The Internet of Things is closer to being implemented than the average person would think. Most of the necessary technological advances needed for it have already been made, and some manufacturers and agencies have already begun implementing a small-scale version of it. The main reasons why it has not truly been implemented is the impact it will have on the legal, ethical, security and social fields. Workers could potentially abuse it, hackers could potentially access it, corporations may not want to share their data, and individual people may not like the complete absence of privacy. For these reasons, the Internet of Things may very well be pushed back longer than it truly needs to be. From this research it can be summarized that both CoAP and MQTT are having their advantages in different use cases. MQTT is more suitable for IoT messaging and nodes with no power constraints would prefer MQTT. CoAP on other hand has efficient power management and is more applicable in utility field area networks. Both have tree architectures. Depending on the hardware of the IoT node and data requirements either MQTT or CoAP can be used as both are significantly lightweight M2M protocols.

When data flows in from one client to multiple nodes, MQTT is much suited for such environments and CoAP for the rest. The experiments and implementations conducted in this research yielded results which make it clear that CoAP is reliable due to the re-transmission process and less packet loss while delay stays high. IoT communication mechanism has been more important component in the whole system than the system itself. New technologies are coming up which would remove the existing barriers and allow more robust communications to happen. There are several other factors which can actually contribute to the growth of IoT and advancement in its field. For example, the battery technology has been stagnant for a long time after the usage of Lithium Ion and Lithium Polymer batteries. They do have size and power limitations and may satisfy partial needs of an IoT module/node.

If the battery technologies improve on and provide more power in a small form factor, this will help the devices to choose from more reliable protocols or modes of communication even though power consumption is little bit higher.

IoT will unveil a whole new era of data exchange, interaction and wireless devices embedded in every corner of the world.

CHAPTER 9

FUTURE WORK

There is every scope of improvising the work done in this research. This chapter will discuss the further improvements and work that can be done to make this research useful. Current experiments for analysis were carried on a Raspberry pi mode B+ and ESP8266 also known as NodeMCU. The development boards used have very less CPU power and memory which were considered along with the analysis results obtained. In the future lots of new IoT hardware is coming which will be capable of housing more wireless modes of communication like Bluetooth, Wi-Fi, NFC, RF, Infrared all in one SOC (System on Chip). This will reduce the size of the chipset and will make the support wider. It is estimated for every existing manual operation/system to be monitored or automated by an IoT node. From ranging to sensors equipped in a manufacturing plant to a smart dustbin which informs the user when it's time to empty it, IoT will creep into every device and collect data. This data will be extremely useful in marketing strategies and business solutions.

This research can be extended by creating a hub of multiple IoT nodes and develop a prediction algorithm for the behavior of the nodes. This can be achieved by applying artificial intelligence along with machine learning algorithms. This was partially implemented in R tool where the required values of support and confidence are plotted. Another extension of this research would be to build a web interface to monitor all the nodes/sensors live with a given refresh frequency. A web service would pull data from all nodes at regular intervals and then would update the visualizations for the user in real time. This future work will help analyze the real-time problems that can occur during the data collection process, maintaining the data on a database and giving the application a seamless flow between data collection and representation to the user.

The nodes which were used in this research were temperature sensor, humidity sensor and few visual indicators (LED). This can be upgraded with wide variety of sensors which can give multi-tuple outputs like an accelerometer or a 9-axis IMU with gyro sensor on it. This will allow testing the speed and ability of a protocol to handle multi-tuple data for every cycle. Overall these are the few things to be considered to improve the current research that was done on IoT communication and protocol analysis.

BIBLIOGRAPHY

[1] Keertikumar M, Shubham M, R. M. Banakar, “Evolution of IoT in smart vehicles: An overview”, 2015 International Conference on Green Computing and Internet of Things.

[2] Abdessamad Mektoubi, Hicham Lalaoui Hassani, Hicham Belhadaoui, Mounir Rifi, Abdelouahed Zakari, “New approach for securing communication over MQTT protocol A comparaison between RSA and Elliptic Curve” 2016 Third International Conference on Systems of Collaboration (SysCo).

[3] Pavel Smutný, “Different perspectives on classification of the Internet of Things”, 2016 17th International Carpathian Control Conference (ICCC).

[4] Sanaah Al Salami, Joonsang Baek, Khaled Salah, Ernesto Damiani, “Lightweight Encryption for Smart Home” 2016 11th International Conference on Availability, Reliability and Security (ARES).

[5] Soumya Kanti Datta, Christian Bonnet, “Easing IoT application development through DataTweet framework”, 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT).

[6] Xinlei Wang, Jianqing Zhang, Eve M. Schooler, Mihaela Ion, “Performance evaluation of Attribute-Based Encryption: Toward data privacy in the IoT”, 2014 IEEE International Conference on Communications (ICC).

[7] Jia Guo, Ing-Ray Chen, Jeffrey J. P. Tsai, Hamid Al-Hamadi, “A hierarchical cloud architecture for integrated mobility, service, and trust management of service-oriented IoT systems”, 2016 Sixth International Conference on Innovative Computing Technology.

[8] Jarkko Kuusijärvi, Reijo Savola, Pekka Savolainen, Antti Evesti, “Mitigating IoT security threats with a trusted Network element”, 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST).

[9] Hiroshi Kawazoe, Daisuke Ajitomi, Keisuke Minami, “Large-scale and real-time remote control architecture for home appliances”, 2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE).

[10] Paolo Bellavista, Alessandro Zanni, “Towards better scalability for IoT-cloud interactions via combined exploitation of MQTT and CoAP” 2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI).

[11] Xiangyu Zhang, Rajendra Adhikari, Manisa Pipattanasomporn, Murat Kuzlu, Saifur Rahman Bradley, “Deploying IoT devices to make buildings smart: Performance evaluation and deployment experience”, 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT).

[12] Vyacheslav Kharchenko, Maryna Kolisnyk, Iryna Piskachova, Nikolaos Bardis, “Reliability and Security Issues for IoT-based Smart Business Center: Architecture and Markov Model”, 2016 Third International Conference on Mathematics and Computers in Sciences and in Industry (MCSI).

[13] Zhaozong Meng, Zhipeng Wu, Cahyo Muvianto, John Gray, “A Data-Oriented M2M Messaging Mechanism for Industrial IoT Applications”, IEEE Internet of Things Journal-2017.

[14] Diana Cecilia Yacchirema Vargas, Carlos Enrique Palau Salvador, “Smart IoT Gateway for Heterogeneous Devices Interoperability”, IEEE Latin America Transactions-2016.

[15] Gustavo A. da Costa, João H. Kleinschmidt, “Implementation of a wireless sensor network using standardized IoT protocols”, 2016 IEEE International Symposium on Consumer Electronics (ISCE).

[16] Hiro Gabriel Cerqueira Ferreira, Edna Dias Canedo, Rafael Timóteo de Sousa, “IoT architecture to enable intercommunication through REST API and UPnP using IP, ZigBee and Arduino”, 2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications.

[17] Zoran B. Babovic, Jelica Protic, Veljko Milutinovic, “Web Performance Evaluation for Internet of Things Applications”, IEEE Access-2016.

[18] Tetsuya Yokotani, Yuya Sasaki, “Comparison with HTTP and MQTT on required network resources for IoT”, 2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC).

[19] Krešimir Grgić, Ivan Špeh, Ivan Heđi, “A web-based IoT solution for monitoring data using MQTT protocol”, 2016 International Conference on Smart Systems and Technologies (SST).

[20] Sanket Thakare, Akshay Shriyan, Vikas Thale, Prakash Yasarp, Keerthi Unni, “Implementation of an energy monitoring and control device based on IoT”, 2016 IEEE Annual India Conference (INDICON).

[21] Ahmed Imteaj, Tanveer Rahman, Muhammad Kamrul Hossain, Saika Zaman, “IoT based autonomous percipient irrigation system using raspberry Pi”, 2016 19th International Conference on Computer and Information Technology (ICCIT).

[22] Amélie Gyrard, Pankesh Patel, Amit Sheth, Martin Serrano, “Building the Web of Knowledge with Smart IoT Applications”, IEEE Intelligent Systems-2016.

[23] H. Rashidzadeh, P. S. Kasargod, T. M. Supon, R. Rashidzadeh, M. Ahmadi, “Energy harvesting for IoT sensors utilizing MEMS technology”, 2016 IEEE Canadian Conference on Electrical and Computer Engineering.

[24] G. V. Vivek, M. P. Sunil, “Enabling IOT services using WIFI - ZigBee gateway for a home automation system”, 2015 IEEE International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN).

[25] Shinho Lee, Hyeonwoo Kim, Dong-kweon Hong, Hongtaek Ju, “Correlation analysis of MQTT loss and delay according to QoS level”, The International Conference on Information Networking 2013 (ICOIN).

[26] Amr El Mougy, Aly El-Kerdany, “Reliable data transfer for collecting data from BLE sensors using smartphones as relays to the cloud”, 2016 11th International Conference on Computer Engineering & Systems (ICCES).

[27] Jianqi Shi, Xin Ye, Liangyu Chen, Pei Zhang, Ningkang Jiang, “ESF - An Extensive Service Foundation from Internet of Things Perspective”, 2012 IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops.

[28] Christian Nemec, Julian Wölfle, Maximilian Nitzsche, Jörg Roth-Stielow, “Handling of disturbance variables within a multi-phase interleaved-switched inverter by a discrete-time decoupling network”, 2013 15th European Conference on Power Electronics and Applications (EPE).

[29] Meena Singh, M. A. Rajan, V. L. Shivraj, P. Balamuralidhar, “Secure MQTT for Internet of Things (IoT)”, 2015 Fifth International Conference on Communication Systems and Network Technologies.

[30] Hyun-Chul Jo, Hyun-Wook Jin, “Adaptive Periodic Communication over MQTT for Large-Scale Cyber-Physical Systems”, 2015 IEEE 3rd International Conference on Cyber-Physical Systems, Networks, and Applications.

[31] Tomasz Kalbarczyk, Christine Julien, “XD (Exchange-Deliver): A Middleware for Developing Device-to-Device Mobile Applications”, 2016 IEEE/ACM International Conference on Mobile Software Engineering and Systems (MOBILESoft).

[32] Lili Zhang, Shusong Yu, Xiangqian Ding, Xiaodong Wang, “Research on IOT RESTful Web Service Asynchronous Composition Based on BPEL”, 2014 Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics.

[33] Navjot Kaur Walia, Parul Kalra, Deepti Mehrotra, “An IOT by information retrieval approach: Smart lights controlled using WiFi”, 2016 6th International Conference - Cloud System and Big Data Engineering.

[34] Virendra Gupta, Jayaraghavendran, “Invited Talk: IoT Protocols War and the Way Forward”, 2015 28th International Conference on VLSI Design.

[35] Shikhar Bahl, Peeyush Chandra, Vandana Rathore, Alka Shukla, Akash Garg, “Wireless ethernet for IoT: A case study”, 2016 10th International Conference on Intelligent Systems and Control (ISCO).

[36] Link: https://nest.com/

[37] Dusan Zivkov, Daniel Kurtjak, Mladen Grumic, “GUI rendering engine utilizing Lua as script”, 2015 IEEE 1st International Workshop on Consumer Electronics (CE WS).

[38] Yanping Wang, Zongtao Chi, “System of Wireless Temperature and Humidity Monitoring Based on Arduino Uno Platform”, 2016 Sixth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC).

[39] Link: https://thingspeak.com/

[40] Tyson T. Brooks, “Wearable IoT Computing: Interface, Emotions, Wearer'S Culture, and Security/Privacy Concerns”, Cyber-Assurance for the Internet of Things-2017.

[41] P. Rajalakshmi, S. Devi Mahalakshmi, “IOT based crop-field monitoring and irrigation automation”, 2016 10th International Conference on Intelligent Systems and Control (ISCO).

[42] Arwa Alrawais, Abdulrahman Alhothaily, Chunqiang Hu, Xiuzhen Cheng, “Fog Computing for the Internet of Things: Security and Privacy Issues”, IEEE Internet Computing.

[43] Peter Cope, Joseph Campbell, Thaier Hayajneh, “An investigation of Bluetooth security vulnerabilities”, 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC).

[44] Evgeny Tsimbalo, Xenofon Fafoutis, Robert J. Piechocki, “CRC Error Correction in IoT Applications”, IEEE Transactions on Industrial Informatics-2017.

[45] Andreas Pfeiffer, Maria Grazia Pia, “Data analysis with R in an experimental physics environment”, 2013 IEEE Nuclear Science Symposium and Medical Imaging Conference (2013 NSS/MIC).

[46] Dongyu Feng, Ligu Zhu, Lei Zhang, “Research on improved Apriori algorithm based on MapReduce and HBase”, 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC).

[47] Kishore Kumar Reddy N. G., Rajeshwari K., “Interactive clothes based on IOT using NFC and Mobile Application”, 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC).

[48] Garrett S. Rose, “Security meets nanoelectronics for Internet of things applications”, 2016 International Great Lakes Symposium on VLSI (GLSVLSI).

[49] Bitan Banerjee, Amitava Mukherjee, Mrinal Kanti Naskar, Chintha Tellambura, “BSMAC: A Hybrid MAC Protocol for IoT Systems”, 2016 IEEE Global Communications Conference (GLOBECOM).

[50] Shaibal Chakrabarty, Monica John, Daniel W. Engels, “Black routing and node obscuring in IoT”, 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT).

[51] Zhonglei Gu and Yang Liu, “Scalable Group Audio-Based Authentication Scheme for IoT Devices”, 2016 12th International Conference on Computational Intelligence and Security.

[52] Azhar Syed, R. Mary Lourde, ” Hardware Security Threats to DSP Applications in an IoT Network”, 2016 IEEE International Symposium on Nanoelectronic and Information Systems.

CURRICULUM VITAE

Graduate College

University of Nevada, Las Vegas

Priyanka Thota

Degrees:

Bachelor of Technology in Computer Science, 2015

Rajiv Gandhi University of Knowledge Technologies

Master of Science in Computer Science, 2017

University of Nevada, Las Vegas

Thesis Title: Implementation and Analysis of Communication Protocols in Internet of Things

Thesis Examination Committee:

Chairperson, Dr. Yoohwan Kim, Ph.D.

Committee Member, Dr. Ajoy K. Datta, Ph.D.

Committee Member, Dr. Juyeon Jo, Ph.D.

Graduate College Representative, Dr. Venkatesan Muthukumar, Ph.D.