Development of an automation tool for internet of things

Submitted By

Vikas Shah

13MCEN15

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2015

Development of an automation tool for internet of things

Major Project

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology in Computer Science and Engineering (Networking Technologies)

Submitted By

Vikas Shah

(13MCEN15)

Guided By

Prof. Rupal Kapdi

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2015

Certificate

This is to certify that the major project entitled **"Development of an automation tool for internet of things"** submitted by **Vikas Shah (Roll No: 13MCEN15)**, towards the partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science and Engineering (Networking Technologies) of Institute of Technology,Nirma University, Ahmedabad, is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Prof. Rupal Kapdi
Guide & Associate Professor,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Prof. Gaurang Raval
Associate Professor,
Coordinator M.Tech - CSE
Institute of Technology,
Nirma University, Ahmedabad

Dr. Sanjay Garg
Professor and Head,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr K Kotecha
Director,
Institute of Technology,
Nirma University, Ahmedabad

Statement of Originality

I, **Vikas Shah**, Roll. No. **13MCEN15**, give undertaking that the Major Project entitled "**Development of an automation tool for internet of things**" submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Computer Science & Engineering (Networking Technologies)** of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

_____

Signature of Student

Date:

Place:

Endorsed by

Guide Name

(Signature of Guide)

Acknowledgments

It gives me immense pleasure in expressing thanks and profound gratitude to **Prof. Rupal Kapdi**, Associate Professor, Computer Science Department, Institute of Technology, Nirma University, Ahmedabad for his valuable guidance and continual encouragement throughout this work. The appreciation and continual support he has imparted has been a great motivation to me in reaching a higher goal. His guidance has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come.

It gives me an immense pleasure to thank **Dr. Sanjay Garg**, Hon'ble Head of Computer Science and Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his kind support and providing basic infrastructure and healthy research environment.

A special thank you is expressed wholeheartedly to **Dr K Kotecha**, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for the unmentionable motivation he has extended throughout course of this work.

I would also thank the Institution, all faculty members of Computer Engineering Department, Nirma University, Ahmedabad for their special attention and suggestions towards the project work.

See that you acknowledge each one who have helped you in the project directly or indirectly.

- Vikas Shah

13MCEN15

Abstract

Internet of things allows devices to communicate without human intervention and to connect devices or machine to internet so we can remotely monitor the status of devices and useful information and can control devices remotely also. internet of things provides protocols that allows devices to communicate remotely and also we can access information from devices and can control remotely too. COAP and MQTT protocols that helps constrained devices to live longer life and device to device communication also. Internet of things allows devices to connect to internet so security plays major role here. and DTLS (Datagram transport layer security) over UDP provides security to this tiny devices and can easily connect to internet. for this tiny devices power management is also a major thing because this needs minimal processing in constrained nodes so this COAP protocols allows this tiny devices to save power by processing minimal header format instead of big HTTP header that saves lot of cost in memory and power in constrained devices.

Abbreviations

IOT	Internet Of Things.
COAP	Constraint application protocol.
MQTT	Message queue telemetry transport.
DTLS	Datagram transport layer security

Contents

List of Figures

Chapter 1

Introduction

1.1 Internet Of Things

Internet of things describes a vision where object becomes part of internet. Where every objects are uniquely identified and accessible to the network its position and status known. Internet of things is aimed to provide communication between object-object and human-object. As day by day number of devices is increasing and has capability to sense and actuate or perform some action All the devices are connected to internet and has capability to sense and send data over network. We can get this data and make use of the data to predict or take some action based on the collected data. The definition of thing could be object such as mobile phones, tablets and home appliances also. These devices or things are connected to through network each providing data and information and some, even services. The Internet of things vision enhances connectivity from any-time, any-place for any one. Once these devices are connected to network then more and more data and services will be available which helps for different types of applications. Internet of things creates applications in variety of field such as agriculture, logistics, home automation etc.

As internet growing rapidly and many devices are connecting day by day. So we can use this connectivity and can make those individual objects to communicate that will share some data and can process otherwise can give to user or can store somewhere where user can use that data to make decision. Internet of things connecting many devices so we need strong connectivity and security. Internet of things has small embedded device that must consume less power and with low cost also. To provide and to overcome those problems we need to research so we can come up with a solution that allows users to develop applications for internet of things. The idea of internet of things came based on RFID tags in which each RFID tag is uniquely identifiable and information is embedded to each RFID tag so we can identify each RFID tag and information embedded into that. Same for internet of things in which all objects are uniquely identifiable in the network and has some information where we can identify each object in network and can get some information from that particular object. Devices are capable to sense data that has built in sensors and processing power also has storage capability increased and size is reduced. These all devices have sensors connected which sense data and send data to cloud where we can process all the data and make some conclusion. [1]

1.1.1 Architecture

Todays internet is using TCP/IP protocol for communication between network hosts now same devices in IOT will use TCP/IP network for communication with different devices. That will increase more traffic in network but with advancement in device capability it will also increase storage and network capability. The basic architecture for IOT

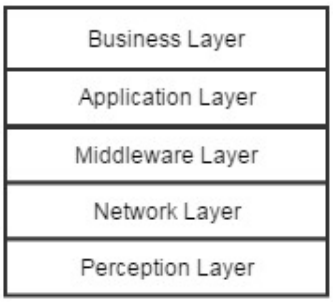

Figure 1.1: Internet of things architecture

As shown in 1.1 IOT consist of five layers and each layer has its own role to support IOT. These layers are [2]

- Perception Layer: The perception layer is also known as Device layer. It consist of physical device and sensors. The sensors are of any type that can be location, temperature, pressure, acceleration etc. This sensors sense the value based on type of sensor and they will pass it to network layer from where they can transmit it to network. The collected information is passed to network layer where network layer provides security to send data to cloud. [3]

- Network Layer: Network layer is also called as transmission layer this layer is used to transfer data collected by sensor to processing device through wired or wireless technology. To transfer information interface can be WIFI, Bluetooth, ZigBee, 3G etc. this layer is connectivity layer between sensor devices and processing devices where we can process all the data and can make some conclusion.

- Middleware Layer: The middleware layer is also called as storage layer. In which all the data collected from network this layer will store it into database or cloud from where application can use the data stored in database.

- Application layer: Application layer has applications where they can use data stored in database or at middleware layer and can make use of the data. For example the application implemented in IOT can be smart health, smart home, smart city and intelligent transportation. This layer collects data from cloud or storage and can give data to user.

- Business Layer: This layer is responsible for the management of overall IOT system including the application and services. It builds business models with graph or flowchart of whole system. Based on the analysis of results can take the decision and can take future actions and business strategies.

1.1.2 IOT challenges

There are various challenges are there in respective of IOT as information is passed from network and anybody can access information. Various challenges in IOTs are [2]

- Naming and Identity management The IOT will connect billion of devices or objects and each should be uniquely identifiable in network. Each object/sensor has unique identity over the internet. The efficient naming and identity management system required that can dynamically assign and manage unique identity for such a large number of objects.

- Standardization There are different manufacturer involved in manufacturing different devices with their own technology where in some cases this devices cannot able to communicate with device that is manufactured by different manufacture so the standardization between all the devices is needed so all the devices can communicate easily.

- Data confidentiality and encryption Sensors will sense the data and send it to network so confidentiality of information is very important where we can use some standard encryption/decryption to encrypt the data and send over network and other device can decrypt it.

- Network Capability The challenge regarding network capability is there are many sensors and devices connected with network and the data from sensor device will send through wired or wireless interface. The transmission system or network should be able to collect all the data from sensors and make sure that no data loss from any sensor due to network congestion.

- Low power The main challenge for IOT is low power of devices as embedded devices used in IOTs are deployed many places and that has limited power capacity in this case its very important to save power whenever its possible. So there is need of mechanism where we can power off the devices when there is no need of power and can power up again whenever needed.

Chapter 2

Literature Survey

2.1 COAP (Constrained application protocol)

There are many communication protocols are available to communicate with different devices currently there are many communication protocols available such as MQTT and COAP. COAP (constrained application protocol) is used for low power and low memory embedded devices where it can be used for communication instead of HTTP. Currently there is HTTP protocol available with request and response paradigm but HTTP has many features and more footprint. And HTTP runs over TCP where TCP will need more resources due to three way handshake and many more complex mechanism. Now for low power embedded devices there is no need of this heavy protocols and we can optimize it to run over TCP. TCP also contains congestion and reliability mechanism so it will take more time and more resources. COAP is easy protocol implementation for communication between embedded devices. There is packet format that can support both reliability and no reliable applications.

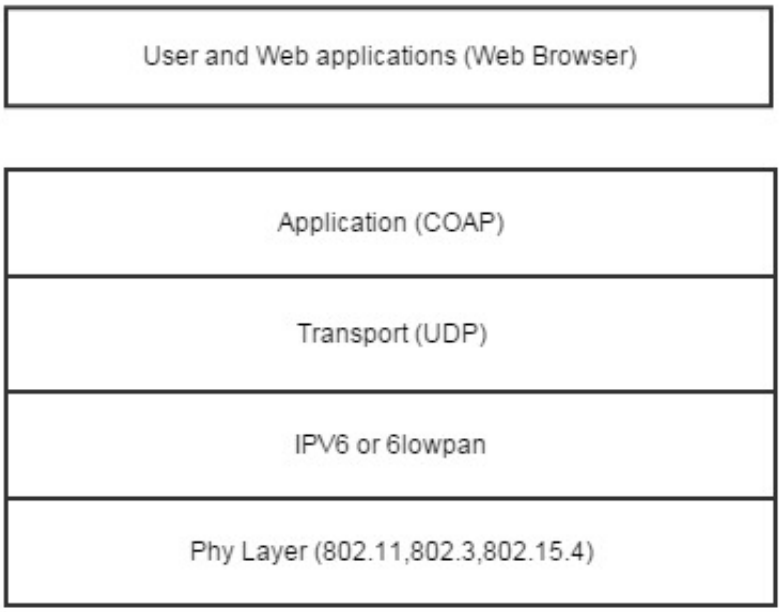

Figure 2.1: COAP Architecture

As COAP is a Restful web transfer protocol for use with constrained networks. COAP uses client/server model of approach same as HTTP. It is designed for constrained networks with low overhead and lower footprint. Some points for COAP that makes better protocol compared to HTTP is [4]:

- COAP runs over UDP (User datagram protocol) that helps to avoid costly TCP handshake before data transmission.
- COAP protocol is only 4 byte header and provides reliable transfer and no reliable transfer as it uses four type of messages
  - Confirmable: This type of message provides reliability over UDP. Where in some applications we need to provide acknowledgement also in this case client will send the packet by setting this message type and server will give acknowledgement.
  - Non-Confirmable: This type of messages provides no reliability and used for applications where there is no need for acknowledgement or reliability in this case client can send the no confirmable message by setting message type.
  - Acknowledgement/Reset message: This type of message is used to provide acknowledgement back to client for confirmable messages.
- COAP has minimal header format that saves lot of power for constrained nodes compare to running HTTP in that constraint nodes.
- COAP provides both reliability and non-realiability support that allows COAP to use in both kind of use case or applications.

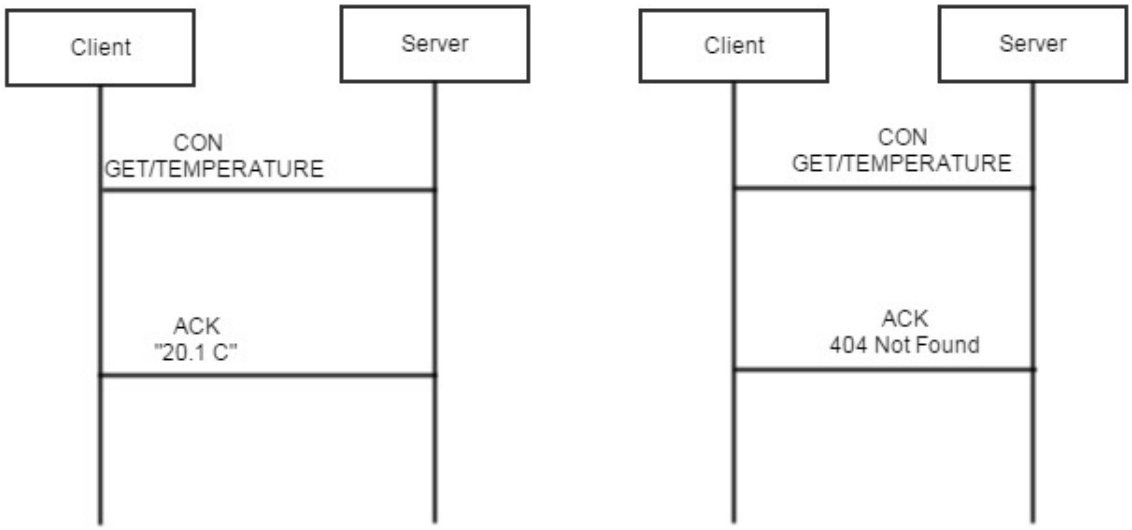

Figure 2.2: COAP Communication

2.2 DTLS (datagram transport layer security)

To provide security above UDP layer DTLS (Datagram transport layer security) plays major role where security is very important for small embedded devices. The basic idea to design DTLS is to provide the security over UDP. We cannot use TLS there because datagrams may be lost or reordered. And TLS has no any internal facility to provide reliability. Unreliability creates problems for TLS at two levels [5]

- TLS dont allow independent decryption of individual records. Because integrity checks depends on sequence no if record N is not received then the integrity check on record (N+1) will be based on the wrong sequence no and it will be failed.
- TLS handshake assumes that handshake messages are delivered reliably and it breaks if those messages are lost.

DTLS protocol is composed of two layers lower layer called as DTLS Record protocol layer which provides connection security that has two basic properties i) The connection is private by symmetric encryption ii) the connection is reliable by adding message integrity so no one can temper the data. The protocol contains four higher level protocols

- DTLS Handshake Protocol : This protocol is used for negotiating security parameters that will be used for encrypting actual content of message.
- DTLS change cipher spec protocol : The change cipher protocol exits to signal transition in ciphering strategies. It is used to notify the receiving party that subsequent records will be protected using the negotiated parameters.
- DTLS Alert Protocol : This protocol is used at any time during the handshake and up to the closure of a session, signaling either fatal errors or warnings.
- DTLS Application data : - Application data messages are carried by the record layer and are fragmented, compressed and encrypted based on the current connection state.

DTLS is not fully optimal for constrained network as it was designed for traditional computer networks. For example large buffers are needed to handle message loss by retransmitting the last flight of handshake protocol or to store all the fragments. Another problem is to use x.509 certificates that are used for mutual authentication for both parties. But these certificates are sometime big and can take few kilobytes and that leads to fragmentation in network with small MTU size and that resulting higher chance of retransmission of whole packet if any one flight in packet is missing. So DTLS recommends to use raw public key in which instead of transmitting big chunk of data for certificates we can only send public key and other party can validate the public key.

- Packet loss : TLS works with TCP where it provides reliability but it cannot work with UDP where there is no reliability and messages can be lost and whole TLS handshake may fail. To overcome this problem DTLS comes over UDP to provide security in which client start retransmission timer after sending client hello and wait for hello-verify packet from server if no any packet or hello-verify packet from client then client will assume that either client hello or hello verify request is lost then client will again start retransmitting client hello to server and server will reply with hello-verify response. The server also maintains retransmission and when it will expires the server will also start retransmission.[5]

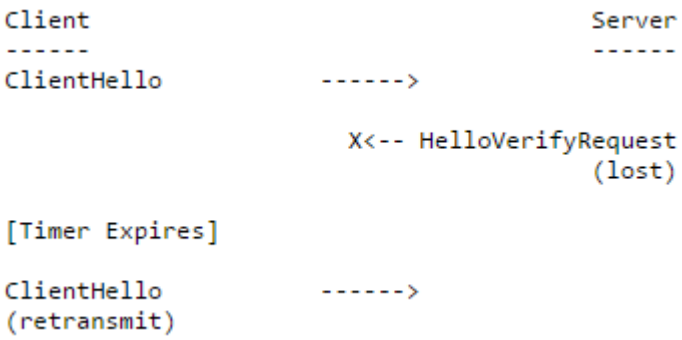

Figure 2.3: Packet loss in DTLS

- Denial of service : to prevent from denial of service DTLS has a mechanism in which after sending client hello first server will send hello-verify request and then again client hello will be sent from client this is also helps to prevent from IP spoofing attack so the server will only maintain state for client only after getting client hello message second time from server. DTLS also support certificate based authentication and PSK (pre shared key) mode where key is already shared between two parties prior. In hello-verify message server generates cookie and send it to client so client will send second client hello by providing same cookie then server will add some state information for clients and start client key exchange. [6]

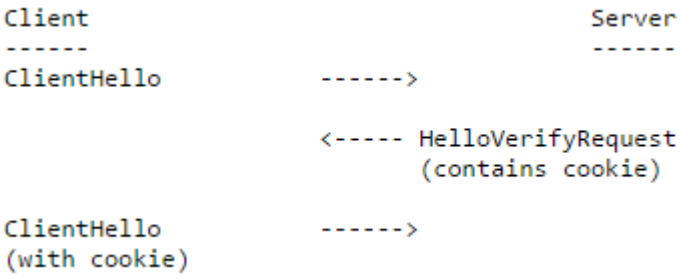

Figure 2.4: Prevent denial of service attack

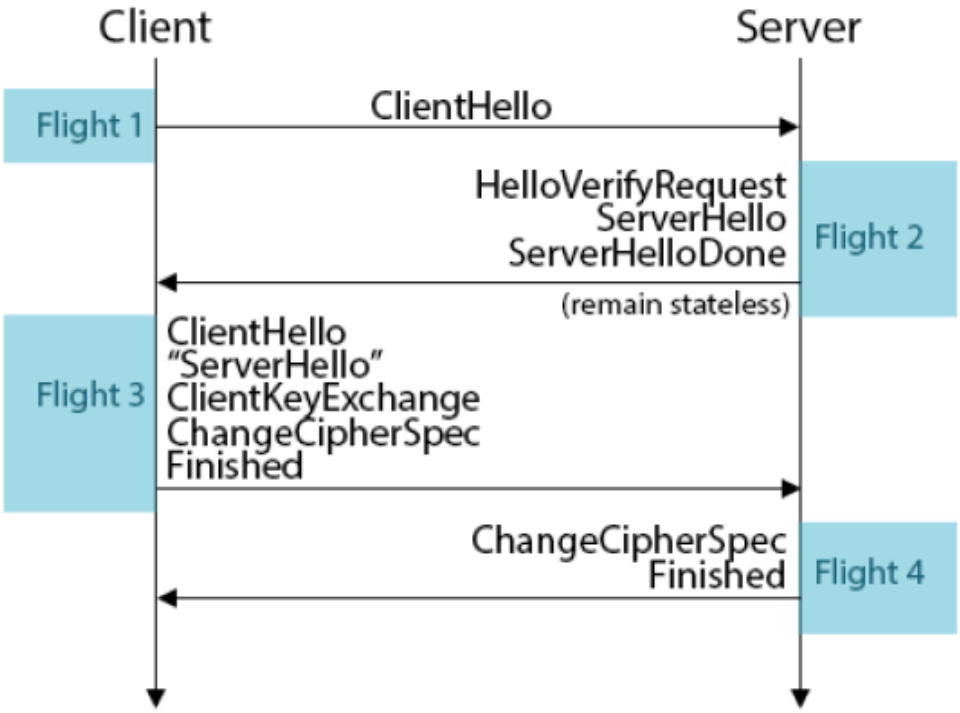

Figure 2.5: DTLS handshake

Chapter 3

Implementation

3.1 Console Tool

To provide access of internet and to support different application internet of things provides WIFI, NFC and Bluetooth APIs. But in embedded system we didnt have any interface through which we can test all the APIs for WIFI, NFC and Bluetooth. To test all the APIs for different radios we need tool through which we can send commands from interface to tool that will send commands to firmware and firmware will give response back and can verify that particular API is working properly or not. There are many challenges as how to make tool through which we can test all the APIs. I came up with packet format through which we can satisfy the requirement. Before there was tool that can help to test all the APIs. But that was a just serial interface in which we can give string text in form of commands but there we need to handle events such as Backspace, Tab that requires extra memory and for embedded systems memory is limited and its difficult to add new APIs testing as there was memory constraint.

- Current Approach With the current approach we can send commands through PUTTY interface to console tool that will reside in embedded system. And console tool will call particular API based on command and that will send commands to firmware and firmware will send response back to console tool and that will show results back to PUTTY again. With the current approach there are problems

  - Console tool will reside in embedded system and manually we need to handle all the events like backspace, tab that will require more memory. For the embedded system there is only limited amount of memory so there is need to optimize console tool.
  - For the testing team there is no scope to automate the testing so there is need for some tool through which we can also automate the whole testing system.

- my approach

  Came up with solution in which we decided to have Perl script interface in PC that will form packet with command and particular arguments and that will be sent through serial interface from Perl to embedded system where there is a console tool that will receive packet and parse packet and will get command and arguments. And based on API id it will call respective APIs that will send request to firmware based on commands and get response back and form packet and sent it to Perl

script where it will decide based on response that API executed successfully or not of API executed successfully then it will form packet for next command and sent it to console interface.

I developed Perl script that will open port and send particular command by forming particular packet to console tool. And console tool will receive all the bytes for particular packet and then start to parse it. After parsing packet based on API id it will map to particular API and call particular API with provided arguments. API will send request to firmware based on API and get response back from firmware and that will form packet and sent it back to Perl script through console application. Major advantage of this approach is we are able to reduce memory required for console tool in embedded system and loading whole console interface tool in embedded we can offload some load or processing to PC so we can reduce memory to console tool. We are also able to do automation as we can write any script to test APIs and can send commands by forming packet from any script and console interface or tool will parse it. So we can automate the whole test framework. At the current stage I have developed Perl script for some commands by forming packet and tested the APIs. In future we are going to develop whole test framework for all the APIs.

  - Learning
    * Perl script to communicate to embedded system and send commands through serial port.
    * Designed the packet format to send from Perl script to embedded device.
    * Got knowledge about execution of API in system and firmware calls.

- Bring up activity : I was involved in new chip bring up activity by porting of some application and solved problems came in porting of applications in new chip. I was involved in supporting of this chip and ported some application in which board can work as both station and AP mode and TCP server and client applications in which board can act as TCP client or server and can communicate through socket with PC. Also ported application scan to show all nearby AP results. Solved problems related to WPS and TCP bidirectional issue.

3.2 Heartbeat Monitor Application

We have interface to connect to cloud through embedded system and WIFI. I developed application in which we can monitor heartbeat of any person that has embedded device with heartbeat sensor that will continuously send data to cloud and android application which can read data from cloud and can show the graph real time. Through this application we can monitor heartbeat of any person or family members remotely. In the application heartbeat sensor will continuously give data to from microcontroller and that will send data to cloud. From cloud android phone has application that will read continuous data from cloud and will plot real time graph through which we can monitor heartbeat remotely.

3.2.1 Learning

- I got the knowledge to interface external sensor with microcontroller through some interface.

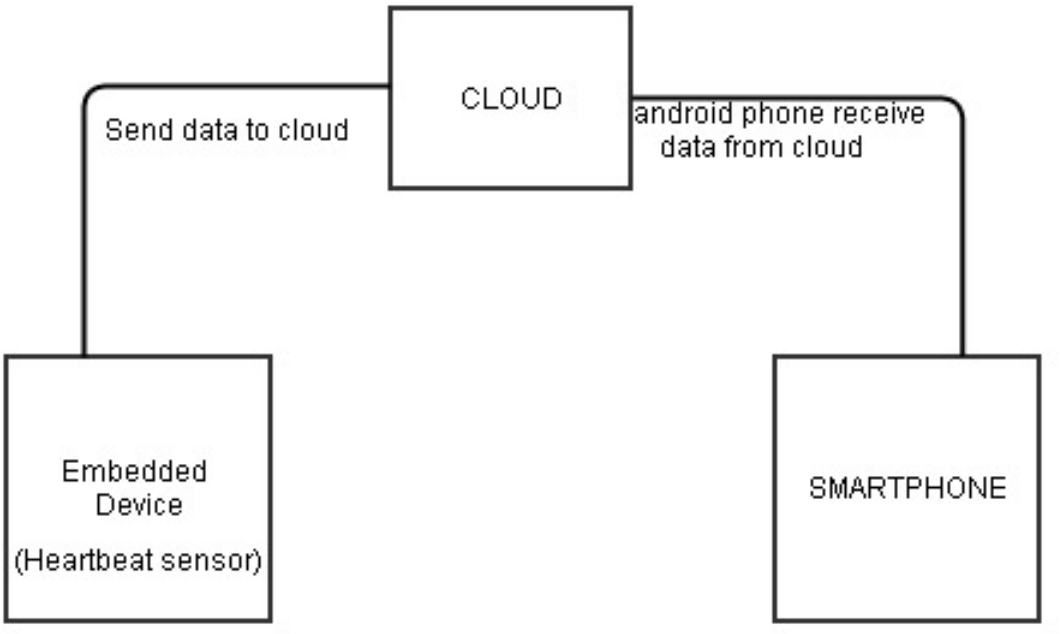

Figure 3.1: Heartrate monitor application

- I got the knowledge to read data from sensor and APIs to send data from microcontroller to cloud.
- Made an android application to get data from cloud and draw the graph real time based on values from cloud.

3.3 COAP (Constrained application protocol)

- Internet of things technology has major concerns over power management and connectivity. For small tiny devices its very important that they consume less power and can respond for long period of time. Now to communicate with this tiny devices we need some protocol that we can give request to the tiny devices and they can respond back with proper response. Currently traditional many protocol are available with which we can communicate with this tiny devices but because of protocol overhead it will consume lot of power for this small devices. So here we need some protocols for constrained environment that can consume less power and can give same features as HTTP. COAP is same protocol as HTTP that give request response mechanism for client and server so client can request for particular resource and server can provide that resource. Here some of the benefits that allows COAP over HTTP to consume less power and provide same functionality. [7]

3.3.1 HTTP v/s COAP

- HTTP runs over TCP where COAP runs over UDP that allows COAP to skip some of the connection establishment sequences.
- Because COAP runs over UDP some of the TCP functionality replicated directed in COAP where we can provide Confirmable (acknowledgement required) request and Non-Confirmable request (No acknowledgement is required).

- Requests and responses are exchanged asynchronously over COAP where in HTTP previously established connection is used.
- HTTP has big HTTP header where COAP has only 4 byte of header.

3.3.2 COAP server features

- Service discovery COAP server has mechanism where client can request for all the available services that are with server and server can respond with all the available services to client and then client can request for particular service. Server has mechanism where it will detect for well-known services request and respond with all the available services. Client can request for discovery of all the resources so server can parse the packet and it will send all the available services registered with server. [8]
- Request for particular service Here server provides all the services and client can perform some operation with particular service. For example home automation server has registered all the available services and client request for all the available services with server and then it can give request for particular service. Here server provides methods for service that you can perform operation. All methods that the server can responds are
  - GET : GET method is available with server that client can give request for particular service. For example if client wants to know that status of TV in house then client will give request with GET method so server will parse request and send the status of TV that it is turn ON or OFF.
  - POST : POST method available with server for particular service where client can give request with method POST and with payload so server can perform the operation. For example client can give POST request with payload 1 for TV as a service and server can parse the request and can turn on the TV.
  - PUT : PUT method allow to update for particular service where you can change that representation of particular service.
  - DELETE : client can also delete some of the services form server if client font want that service with server. Client can send the request with method DELETE so server deletes that service from list of available services with server.

Reliable and Non-Reliable requests [9]

- CON request If client want reliability for particular service and needs acknowledgment that server has received particular request or not at this time after receiving request from client server will parse the request and if its CON request then server will respond for the same by giving acknowledgment.
- NONCON request if client dont want any reliability for request and there is no need of any acknowledgment that server has received request from client or not then client simply set message type as NONCON and server will not give any acknowledgment and just give response with proper payload.

- Separate Response If in some of the cases where client sends the request and server will take more time to process and service that particular request then client will wait for acknowledgment for CON request and it will create lot off retransmissions from client side to prevent this retransmission server will send response as acknowledgment and empty response so client can sense that it will take some more time for server to process and it will not retransmit the packet.

- Piggyback response In some of the cases where response is ready then server will send piggyback response to client where it will combine both data and acknowledgment for request.

Ping request and response Protocol also provides functionality in which client can ping the server and can identify that server is available or not. If server is available then server will give response back with RST message type. And then only client can send the request to server.

Observe Protocol provides option in which client can register with server for interest in particular resource so when value changes for particular service then all the interested client who have all registered with servers are notified with the latest value.

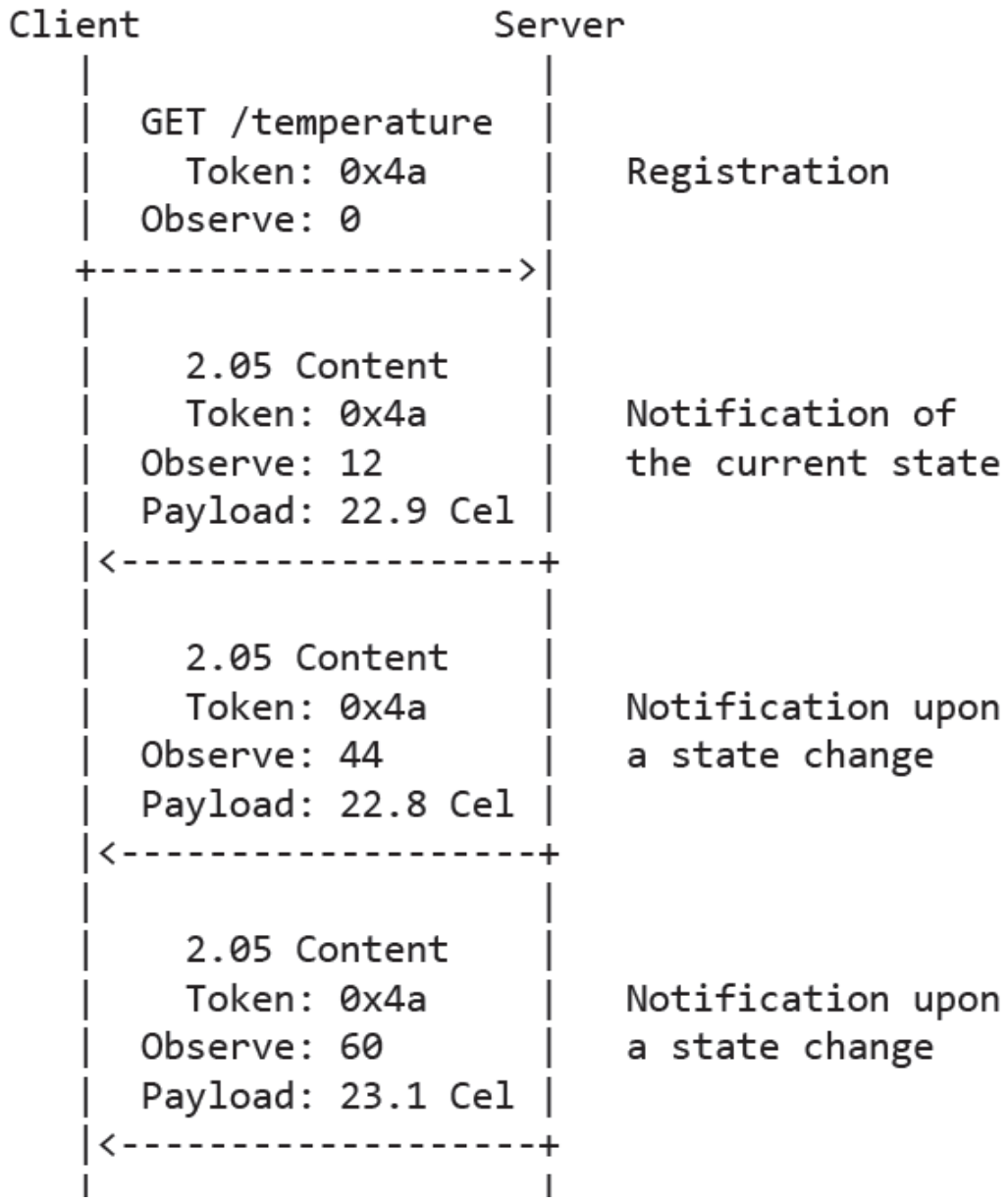

Figure 3.2: Observe for resource

Here first message client registers its interest for particular topic in which client is interested. So client will give the request with method GET and observe option so server will check the option and if Observe = 0 then add client into registered client for particular topic and save token value for that particular client too. If Observe = 1 then deregister or delete client entry from list for particular interested clients if available otherwise dont add it. And server also returns the current value for that particular resource and if client is registered properly and client entry is added to list then server also add option observe with value so client can determine that now he is registered with server for particular topic and whenever state changes for that particular topic then client will be notified with latest value. Now whenever state of any service is changed then server will notify all the registered client for that particular service with latest value and adds observe sequence number so client can identify latest value from server. Sequence number is 24 bit number that will be incremented every time when the state of the service is changed. Server also sends notification with CON type of message so the client will send ACK if client wants

to continue to get updates for particular resource otherwise client will send RST message so server will delete entry for that particular client and remove client from list so after that client will not be notified for any change in status in notifications.

while subscribers are in list of observers then goal of protocol is to keep the resource state updated for all subscriber or observer who observed for the particular resource. Here there will be some latency between the resource state changed and status update to observers but we can not avoid client and server to be out of sync. There are some problems in which COAP notification message can get lost that can cause the client to assume old state of resource until it receives new notification. Also server will assume that client is not interested in this topic now and client don't want to receive any notifications in future so server will delete entry of that particular client from list and server will stop sending notifications so client will assume older state of resource and client has to register its interest again. The protocol addresses this issue as follows :

- Protocol follows best effort approach for sending notification when client state gets changed. but client should see the change in state as soon as possible and it should be also informed about all the possible states.
- It labels notification with a maximum duration that indicates maximum age up to which resource state is considered as latest update. when the age of the notification received reaches the limit then client will not consider that resource representation as a latest notification and client will not use until it receives new notification update.
- The protocol guarantees that if the resource state is not changed then all observer has resource state that will be considered as a latest resource state.

A server that is not able to process the request or not able to add client in registered client list then server will just ignore the registration and process the GET request as usual. The resulting response must not include observe option which signals the client that server is not able to add client into servers registered client list for particular topic and it will not be notified of changes in the state of resource. If the observe option in GET request is set to 1 (deregister) then the server must remove any existing entry with a matching endpoint/token pair from list of observer. And respond with same without including option observe in response.

3.3.3 Modules

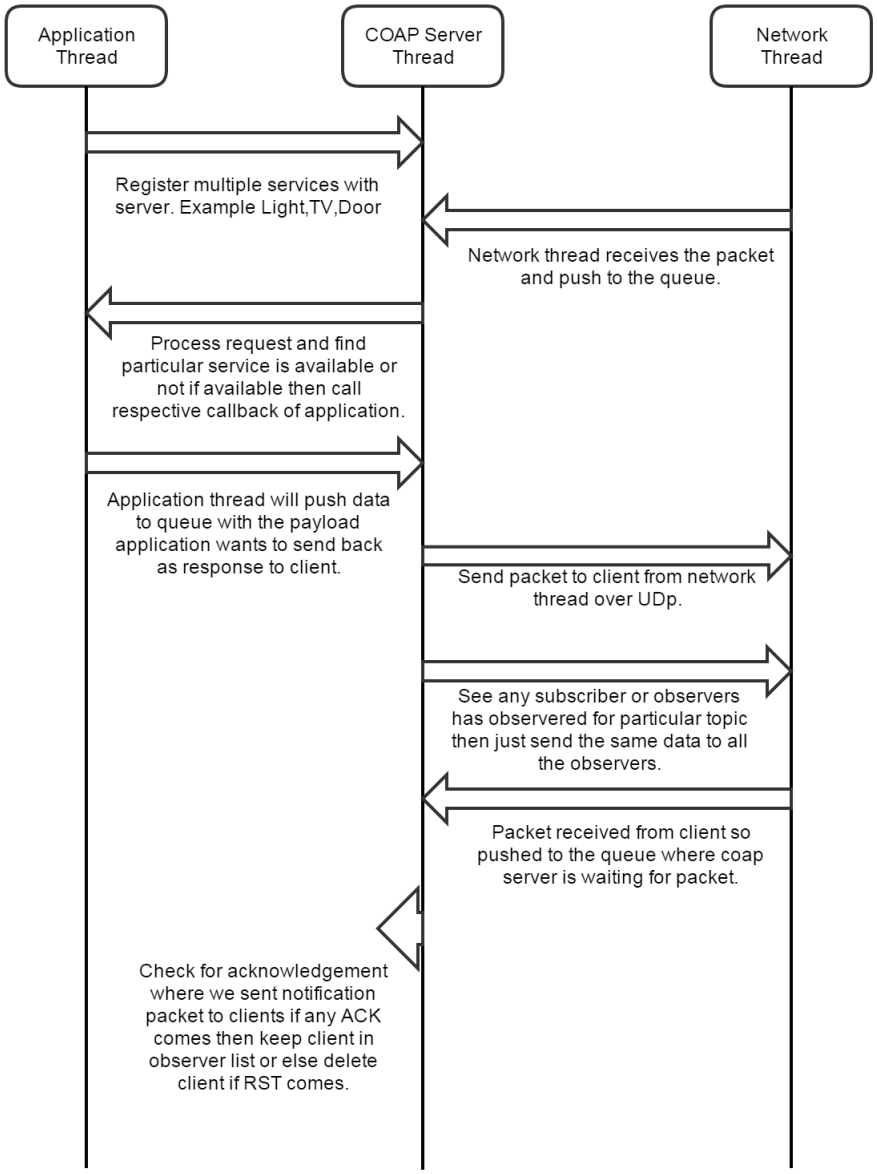

Figure 3.3: Design diagram

I have implemented COAP server that creates thread and waits for request from client. So I have designed module in which user can add any number of services in server side and when client request for particular service the request will be parsed by server and give response to client. If request is type of CON message then server adds ACK in response and give response based on request from client. Server has particular handler for each service which will be available to do task when request comes for particular service from client.

I have designed and implemented module for service discovery, Ping request and response and subscription and notification for particular service. All available services are already registered with server so server can parse the packet and can give proper response. Here COAP server thread is waiting for message in queue once message is

received by network thread then that will push to queue after that COAP server thread start processing packet and call respective callback handler that is already registered. Then after processing request and performing operation application thread will push the data into queue with send event. So server will start for sending data to client and it also checks the observer list that any observer has subscribed for this particular topic if yes then it will send packets to all observers and wait for response if client responds with ACK that means client is still interested in getting any notification from server side if client says RST then it means that client is not interested in now getting notifications for the same topic. Server will try several time if ACK or RST doesnt come and if still no reply is coming then server will delete observer from the list.

3.4 MQTT (Message Queue telemetry transport)

MQTT is a lightweight protocol in which subscriber will subscribe for particular topic with broker and publisher also publishes for same topic then all subscriber will be notified with the published value. Broker also provides some of the functionality for clean session in which it will clear out all the previous messages and clean the session broker also allows publisher to publish messages based on quality of service. Broker also provides option for will message in which if subscriber or publisher got disconnected from broker then all other connected clients will be notified. MQTT runs over TCP so it provides reliability. [10]

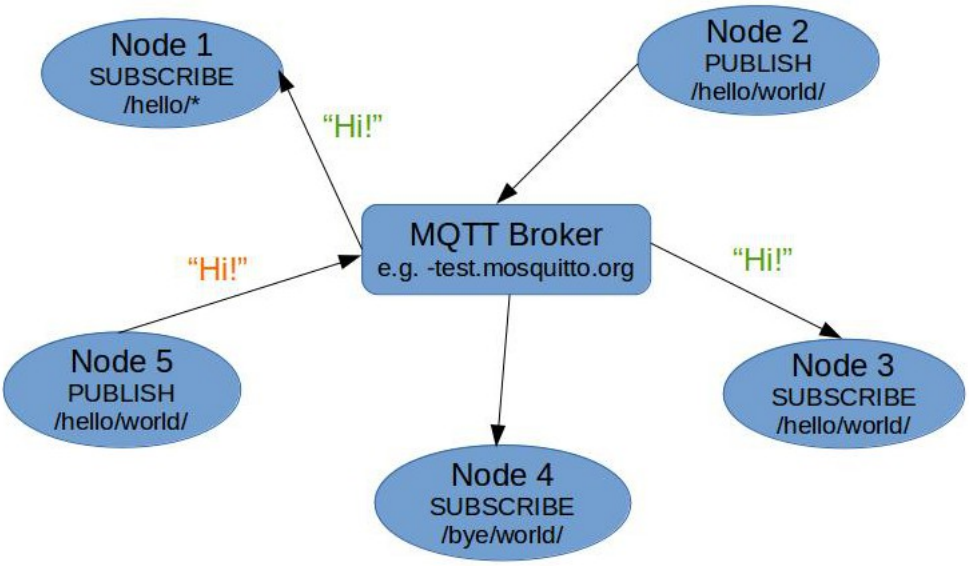

Figure 3.4: MQTT Protocol

- connect message Subscriber or publisher connects with broker with some of the parameters. A first packet from client to server should be connect so client can connect to server and can subscribe for particular topic. Connect packet contains fields for Username, Password, Will retain, Will QOS, clean session etc. Connect message provides username and password so if client specifies correct username and password then client can connect with server and after authorizing only server allows client to connect to server. After sending connect message to broker client will wait for acknowledgement and once acknowledgement received it will start subscribing

or publishing for topic. The fixed header for all the messages is

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (1)				DUP flag	QoS level		RETAIN
	0	0	0	1	x	x	x	x
byte 2	Remaining Length							

Figure 3.5: MQTT Fixed header

- Publish message Publisher also connects to broker first for some topic and send connect message. Once publisher is connected with broker it can publish to that topic. Publish packet also contains some of the fields like DUP (If value of DUP is 0 that means this is the packet sent first time and if value is 1 that means this publish message is duplicate and send again because client didnt get the acknowledgement of previous packet. Publish packet also has field for QOS (Quality of service) which indicates the levels of assurance for delivery of application messages.

- subscribe message Subscriber first connects to broker and then subscribe for particular topic subscriber also has some unique client id and it will have some of the parameters like topic name with requested QOS so broker will maintain all the packets based on QOS levels. If QOS is 0 then it will send traffic as best traffic and if QOS value is 1 then subscriber has to acknowledge this packet but here also duplicate packets may arrive. In QOS 2 broker will ensure that packet should be delivered exactly once and there should not be any duplicate packets. The server must acknowledge packet with SUBACK packet and client waits for packet that publisher publishes. Client can also unsubscribe for particular topic by giving topic name, client id and with proper unsubscribe frame format. So once broker receives this frame broker understands that client or subscriber wants to unsubscribe for particular topic and now he is not interested in getting notification for this particular topic so broker will give acknowledgement or confirmation that client is deleted from the list of subscriber for this particular topic. [11]

- Ping Request and Ping Response: Ping Request packet is sent from client to server. This indicates that client is alive in respect of no transmission of any control packets. Ping response will be sent from server to client that server is also alive and it has received request from client.

- Disconnect Disconnect packet is sent from client to indicate to close the connection to server. after this packet client and server both closes network connection.

- Quality of services

  - QOS 0 : At most Once delivery MQTT supports various QOS levels to provide reliable delivery of any packets. QOS0 indicates best effort delivery of packet. so broker will not wait for any acknowledgment from client and there is no guarantee of packet that it will reach to client or not. when there is no requirement or no need to ensure for delivery of packet then broker can send

packet with QOS level 0. here only 1 packet will arrive to client because there is no retransmission.

– QOS 1 : At least once delivery The quality of service with level 1 ensures that message or packet should reach at least once to client. so first broker will send packet to client and then client will send acknowledgment to broker so broker can ensure that packet is reached to client or not if packet is not reached to client then broker will again retransmit the same packet and wait for acknowledgment but here duplicate packet may arrive to client with duplicate flag set.

Client	Message and direction	Server
QoS = 1 DUP = 0 Message ID = x **Action:** Store message	PUBLISH ---------->	**Actions:** • Store message • Publish message to subscribers • Delete message
Action: Discard message	PUBACK <----------	

Figure 3.6: QOS Support

– QOS 2 : Exactly once delivery This is the highest level of delivery of quality of service where neither loss nor duplication of messages are acceptable. In QOS2 there are two levels of acknowledgment in which when publisher publishes message to broker then client will receive two types of acknowledgment if client is not receiving any acknowledgment from broker then client will again retransmit the publish packet. Must send a publish packet containing this packet identifier with QOS= 2 and DUP =0. Must treat the packet as unacknowledged until it has received the corresponding PUBREC packet from the receiver.

Chapter 4

Conclusion and Future Scope

4.1 Conclusion

Internet of things helps to monitor your things,useful information remotely by connecting sensors to cloud. so technology helps to improve quality of life. with the implementation of protocols like COAP and MQTT it is very useful for the M2M communication where one device communicates to another without human intervention. so main goal to implement COAP is to reduce power consumed by constrained device where HTTP consumes lot more power for this constrained device. so it helps in the lifetime of constrained devices and also allows machine to machine communication without any human intervention. MQTT is very useful in case where our mobile act as publisher and all other devices act as a subscriber so with one click i can communicate to all devices in my home. implementation of DTLS over UDP is also important part here because security is major concern in Internet of things. and TLS only works over TCP so it is very necessary to implement DTLS over UDP that allows to communicate with device by proper authentication and by providing message integrity.

4.2 Future Scope

Implementation of COAP and MQTT protocol allows to do communication with machine to machine without any human intervention. in future to save more power for constrained devices we can implement COAP/HTTP proxy where underlying all the constrained devices will run COAP as a protocol because that has minimal header and saves lot of power for constrained devices compare to HTTP. so underlying network runs COAP protocol and we can design proxy that can act as a gateway and that has capabilities to handle both COAP and HTTP request. so benefit here is that for mobile side we are only using HTTP protocol so from mobile we can request HTTP resource and proxy can map HTTP resource to COAP resource and protocol header conversion. so underlying network only get COAP packet so they can process easily and can give it to proxy now again proxy can convert same COAP header to proper HTTP header conversion and resource mapping and send it to mobile. so here it provides two kind of advantages like mobile no need to worry about COAP prtocol and it can simply send HTTP request and underlying constrained node can only handle COAP so there is no need for constrained nodes to understand HTTP protocol and processing. so it saves lot of power and cost for constrained devices. Proxy can also store some of the reply for resource from constrained devices and if any new request comes then without giving all request to constrained nodes proxy only can handle the request and send response back in HTTP response format. In

future work i also need to implement DTLS over UDP that helps to communicate devices securely over UDP protocol.

References

[1] S. K. Hannes Tschofenig Sye Loong Keoh, "Securing the internet of things : A standardization perspective.," IEEE, IEEE, 2014.

[2] s. u. k. Rifaqat zaheer shahid khan Raullah khan, "Future internet : The internet of things architechture,possible applications and key challenges.," IEEE, IEEE, 2012.

[3] S. K. Noboru Koshizuka Ken Sakamura Takeshi Yashiro, "An internet of things architechture for embedded appliances.," p. 314.

[4] F. B. c. v. D. p. Rodolfo De Paz, Alberola Szyman, "Constrained application protocol for low power embedded networks," pp. 702–707, IEEE, IEEE, 2012.

[5] s. l. k. oscar garcia sandeep Martina Brachmann, "End to end transport security in the ip based internet of things," IEEE, IEEE, 2012.

[6] E. Rescorla, "Datagram transport layer security," pp. 1–104, 2012.

[7] A. P. Zach Shelby Carstern Bormann, "Coap:an application protocol for billions of tiny internet node," pp. 62–67, IEEE, IEEE, 2012.

[8] K. Hartake, "Observing resources in coap," pp. 1–68, December 2014.

[9] Z. Shelby, k.Hartke, and c.Bormann, "Constrained application protocol," pp. 1–236, June 2013.

[10] "Mqtt 3.1," pp. 1–81, October 2014.

[11] X. M. Alvin Valera Colin Keng Dinesh Thangavel, "Performance evalution of mqtt and coap via a common middleware.," pp. 1–6, IEEE, IEEE, 2014.