Home Automation Protocol For IOT

Major Project Report

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology

In

Electronics & Communication Engineering

(Embedded Systems)

By

Darshak Shah

(14MECE21)

Electronics & Communication Engineering Branch
Electrical Engineering Department
Institute of Technology
Nirma University
Ahmedabad-382 481
May 2016

Home Automation Protocol For IOT

Major Project Report

Submitted in partial fulfillment of the requirements for the degree of

Master of Technology
In
Electronics & Communication Engineering
(Embedded Systems)

By

Darshak Shah
(14MECE21)

Under the guidance of

External Project Guide:
Mr. Kausik Sinnaswamy
Manager Software Engineering - Development,
Broadcom Communications Technologies LTD.,
Bangalore.

Internal Project Guide:
Prof. Manisha Upadhyay
Associate Professor (EC Dept.),
Institute of Technology,
Nirma University, Ahmedabad.

Electronics & Communication Engineering Branch
Electrical Engineering Department
Institute of Technology
Nirma University
Ahmedabad-382 481
May 2016

Declaration

This is to certify that

a. The thesis comprises my original work towards the degree of Master of Technology in Embedded Systems at Nirma University and has not been submitted elsewhere for a degree.

b. Due acknowledgment has been made in the text to all other material used.

- Darshak Shah

Disclaimer

NIRMA UNIVERSITY
INSTITUTE OF TECHNOLOGY
NAAC ACCREDITED 'A' GRADE

Certificate

This is to certify that the Major Project entitled **"Home Automation Protocol For IOT"** submitted by **Darshak M Shah (14MECE21)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in Embedded Systems, Nirma University, Ahmedabad is the record of work carried out by him under our supervision and guidance. In our opinion, the submitted work has reached a level required for being accepted for examination.The results embodied in this major project, to the best of our knowledge,haven't been submitted to any other university or institution for award of any degree or diploma.

Date: Place: Ahmedabad

Prof. Manisha Upadhyay	**Dr. N.P. Gajjar**
Guide	Program Coordinator
Dr. P.N.Tekwani	**Dr. P.N.Tekwani**
HOD of EE	Director, IT

Certificate

This is to certify that the Major Project entitled **"Home Automation Protocol For IOT"** submitted by **Darshak M Shah (14MECE21)**, towards the partial fulfillment of the requirements for the degree of Masterr of Technology in Embedded Systems, 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.

Mr. Kausik Sinnaswamy
Manager Software Engineering - Development
Broadcom Communications Technologies Ltd.
Bangalore.

Acknowledgements

It gives me immense pleasure in expressing thanks and profound gratitude to **Prof. Manisha Upadhyay**, Professor, Electronics and Communication 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.

I would like to express a deep sense of gratitude and whole hearted thanks to my project mentor **Mr. Kausik Sinnaswamy** in order to guide me throughout the way. It was his keen interest, encouragement and full cooperation that have made it possible for me to move ahead with the work.

I would like to express sincere thanks to **Dr. N. P. Gajjar**, PG Coordinator of M. Tech. Embedded Systems program for allowing me to undertake this thesis work and for his guidelines during the review process.

I would take this opportunity to express a deep sense of gratitude to **Mr. Sathish Mani** for his cordial support, constant supervision as well as for providing valuable information regarding the project and guidance, which helped me in completing this task through various stages.

Lastly, I thank almighty, my parents, and friends for their constant encouragement without which this assignment would not be possible.

- Darshak Shah
14MECE21

Abstract

In the emerging world of 5G, Internet of Things is one of the advancing technologies. This allows communication between devices, as well as devices and Internet; thereby giving an automated world excluding human intervention. and easier. Remote controlling and monitoring the device status being most important part in the communication is now possible with IOT protocols. HTTP-2, COAP and MQTT are the protocols that help constrained devices to live longer life. Device connectivity to internet carries security factor is playing a key role. TLS (Transport Layer Security) over TCP provides security to these tiny devices along with providing stable connection to internet. The other factor is power management, as minimal processing is required in constrained nodes, HTTP-2 protocol allows these tiny devices to save power by processing minimal header format instead of big HTTP 1.1 header. This saves lot of power of constrained devices. All major IoT communication protocols are briefly described in this thesis and comparison among them for the different applications has been presented. For the comparison of this protocols different data traffic patterns have been used. Results with respect to bandwidth and data rate are also presented for different protocols. Results clearly show that COAP protocol consumes 3 times less bandwidth than the HTTP 1.1 and MQTT generate 9 times less data than the HTTP 1.1. From this thesis one can get an idea to choose suitable IoT protocol for a particular application. Thesis also shows implementations of modules related to some of the IoT protocols.

Contents

List of Figures

Abbreviation Notation and Nomenclature

ALPN	Application Layer Protocol Negotiation
COAP	Constraint application protocol
HTTP	Hyper Text Transfer Protocol
IoT	Internet Of Things
IP	Internet Protocol
RFID	Radio-Frequency Identification
MQTT	Message queue telemetry transport
NPN	Next Protocol Negotiation
RFC	Request for Comments
TLS	Transport layer Security
TCP	Transmission Control Protocol
URL	Uniform Resource Locator

Chapter 1

Introduction

This chapter describes various contexts in which a smart home automation system fits in. It includes the smart home devices and constraint protocols specially developed for the IOT. This chapter contains details about the challenges that the smart home or IOT are facing, also objectives of the thesis. In conclusion the chapter points out the outline of the thesis.

1.1 Internet Of Things

The word IOT was first used by the Kevin Ashton into the 1999 to describe the use case in which every physical object connected to the internet by sensors. Ashton wanted to demonstrate the power of connecting Radio-Frequency Identification (RFID) tags to the Internet. Through which he achieved better performance without need of the human intervention. Today, the Internet of Things has become a famous word to describe the systems which has internet connectivity and computing capability extend to the smaller objects.[3]

Internet of things gives us the power to connect every object to the internet. Where any object, which is a part of the network, can be uniquely identified, controlled and accessed. The object to object and human to object communication can be possible. Every passing day the number of devices which are connected to the internet, the capability to sense and actuate any system from the internet and ability to collect the real-time data and analysis are increasing. We can collect the data and use that data to analysis and predict the behavior of the system. Internet of things creates applications in variety of fields such as agriculture, logistics, home automation etc.

As internet is growing rapidly and many devices are connecting day by day, we can use this connectivity and can make those individual objects to communicate. It will share some data that can be processed otherwise it can be stored on cloud for future predictions. When we talk about the connectivity and data we must have the strong security to maintain data integrity. Internet of things devices are also embedded device which have its own constraints like less power, storage and cost.

Big data movement is the very important factor for the IOT ecosystem. The full coverage of the sensor nodes and the most of the system demands those sensors to sense continuously, which turn into the big massive data in terms of both velocity and volume. Handling of this big velocity and volume data is the real challenge for the data management technologies. Big thing for IOT is the real time data processing, since all of the IOT system should have the ability to react and respond in the real time.[3]

If we take any simple example of the sensing and monitoring IOT application contains 100 sensors, the raw data produced by this type of system can be 4 PB once in 3 years.

1.2 Challenges for IOT

IOT faces many challenges today which need to examine and explore to find the answers related to IOT technology. These include security, privacy, interoperability and standards, legal, regulatory, and rights and emerging economies and development. Some of them are described in brief here.[6]

- **Security:** While we are talking about the all things connected to the internet and remotely controllable one can imagine the risk of the data theft and the insecurity about the personal data integrity. Security is the major research area in the information technology subject. But when it meets with the IOT several other security challenges also may come up. Because many high efficient security algorithms and architecture fails because of the constraint of the IOT system with the low power and low cost requirement. After insuring the data integrity and security people can trust in IOT products and devices. Product should not behave vulnerable because these devices may get integrated with the peoples day to day life. So we have to insure that IOT device is secure from the cyber attack and data theft.

  In the IOT system most of the devices are like self-connected to the internet or uses zero configuration protocol. So for this self-connected devices potentially affects the security and resilience of the Internet globally.

- **Privacy:** The full potential of the Internet of Things depends on strategies that respect individual privacy choices across a broad spectrum of expectations. The data streams and user specificity afforded by IOT devices can unlock incredible and unique value to IOT users, but concerns about privacy and potential harms might hold back full adoption of the Internet of Things. This means that privacy rights and respect for user privacy expectations are integral to ensuring user trust and confidence in the Internet, connected devices, and related services.

- **Interoperability / Standards:** IOT is the grooming area for the connected world and needs to be mature in all terms like hardware, networking protocols, application layer protocols, sensor technology and cloud storage system. So now many people come up with the ideas on each of this subject so currently this area is very fragmented particular for communication protocol. People are evaluating many protocols for the different IOT application. So full interoperability for the different platforms.

1.3 Thesis objective

The main objective and motive of this thesis is to understand the application protocol requirement for the any remote smart home system, specifically which IOT protocol should be used for the specific kind of Application. Based on the result and the analysis we can propose suitable architecture that efficiently collects the home data process it and send it to the cloud for the further processing. There might be the case for one application COAP is the best suitable protocol and other application conventional HTTP is more efficient, so IOT protocol should be evaluated and compare. And finally most efficient IOT communication protocol should be incorporated in the final system.

1.4 Thesis outline

This section gives an overview of the thesis structure and a brief introduction to each chapter.

- **Chapter 2 - Literature Review:** This chapter summarizes the literature work for the Home automation protocols and the IOT ecosystem. In Section 2.1, the introduction about the Smart home definitions and the general smart internet of things infrastructure. Section 2.2 Different communication models for the any general IOT system. Section 2.3 describes the widely used protocols in the field of IOT communication, namely HTTP, HTTP-2, CoAP and MQTT. Finally, section 2.4 concludes the chapter.
- **Chapter 3 - Test Results:** This chapter describes if system is developed by three different protocols (MQTT, CoAP, HTTP) then what could be the behavior of the system with respect to required bandwidth and volume of generated data. By using

companys test setup here describe how one IOT system behave differently for different protocol. From this exercise determined that which component is inefficient with respect to the IOT scenario and can be improve over the period of time.

- **Chapter 4 - Implementation:**This chapter describes how IOT device or embedded device can be on-board to the router and connect to the internet. It also contains implementation flow of how device can find the information about the router such as based on the router's SSID.
- **Chapter 5 - Future work and conclusion:** This chapter mainly talk about the future scope of this thesis and followed by conclusion.

Chapter 2

Literature Survey

This chapter summarizes the literature work for the Home automation protocols and the IOT ecosystem. In Section 2.1, the introduction is about the Smart home definitions and the general smart internet of things infrastructure. section 2.2 describes different communication models for any general IOT system. Section 2.3 describes the widely used protocols in the field of IOT communication, namely HTTP, HTTP-2, CoAP and MQTT. Finally, section 2.4 concludes the chapter.

2.1 Smart Home and Internet of Things

In the current scenario 50 percent of the world population lives in the urban cities and expected that this ratio will increase up to 75 percent till the 2050. There are several reasons for this, including better access to healthcare, entertainment, telecommunication and transportation. The Smart Home term represents the home things which can be accessible and controllable through remote area and make home more lively.

Smart home should be deployed through the sensors, actuators, many IOT communication protocols and the data storage facility. Each of this component can be deployed with the many solutions like for the IOT communication protocol HTTP, COAP, MQTT are there. There is endless list to prototype smart home. This chapter describe some of the most promising IOT protocols used in the actual implementation.

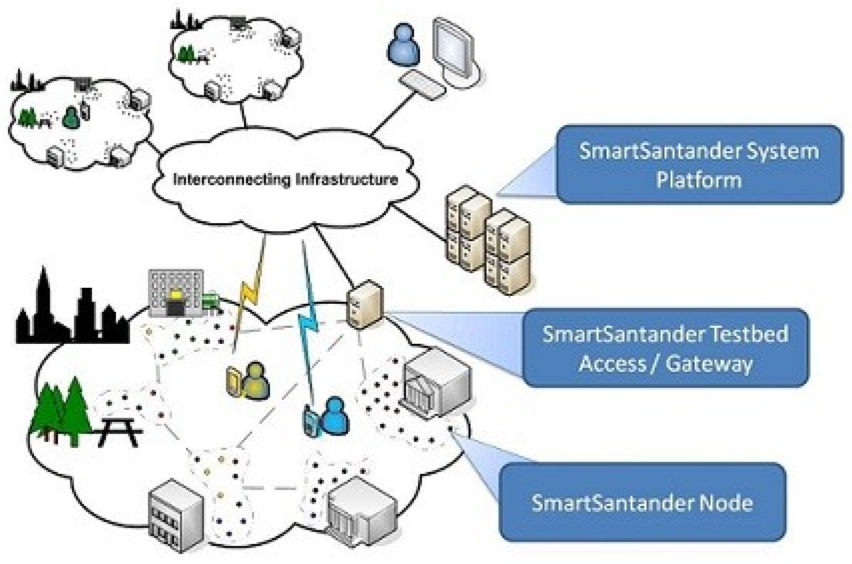

Interconnecting Infrastructure
SmartSantander System Platform
SmartSantander Testbed Access / Gateway
SmartSantander Node

Figure 2.1: General IOT system Architecture[3]

Above fig shows the overall general architecture of the any IOT system and the IOT communication protocol. Below is the some of the key element of the IOT system.

- **Addressing scheme:**in the IOT many devices are connected to the internet directly, so it is very important to identify each device uniquely in the internet. After identifying each device we should be able to access and control the device remotely. Addressing scheme will give power to each device or node to recognize uniquely on to the network. Here address space should have ability to scale with increasing number of devices and network without degradation on the communication channel. Till this date IPv4 is the robust option which is widely used in the internet. However IPv6 is the good replacement for the IPv4. This large number of addressing scheme would be enough to recognize each device in the network uniquely.[3]

- **Connectivity model:** To meet IOT requirement and its constraint. TCP/IP is the best communication model for the any IOT application. TCP/IP is the very much used communication protocol because of the services it provides and it has low-power operation capability, network scalability and meet the constraint requirements. TCP/IP is layered architecture. Every layer in the protocol takes the services of the below layer in the architecture and offer services to the upper layer of the protocol. TCP/IP has physical layer, MAC layer, network layer, transport layer and application layer.

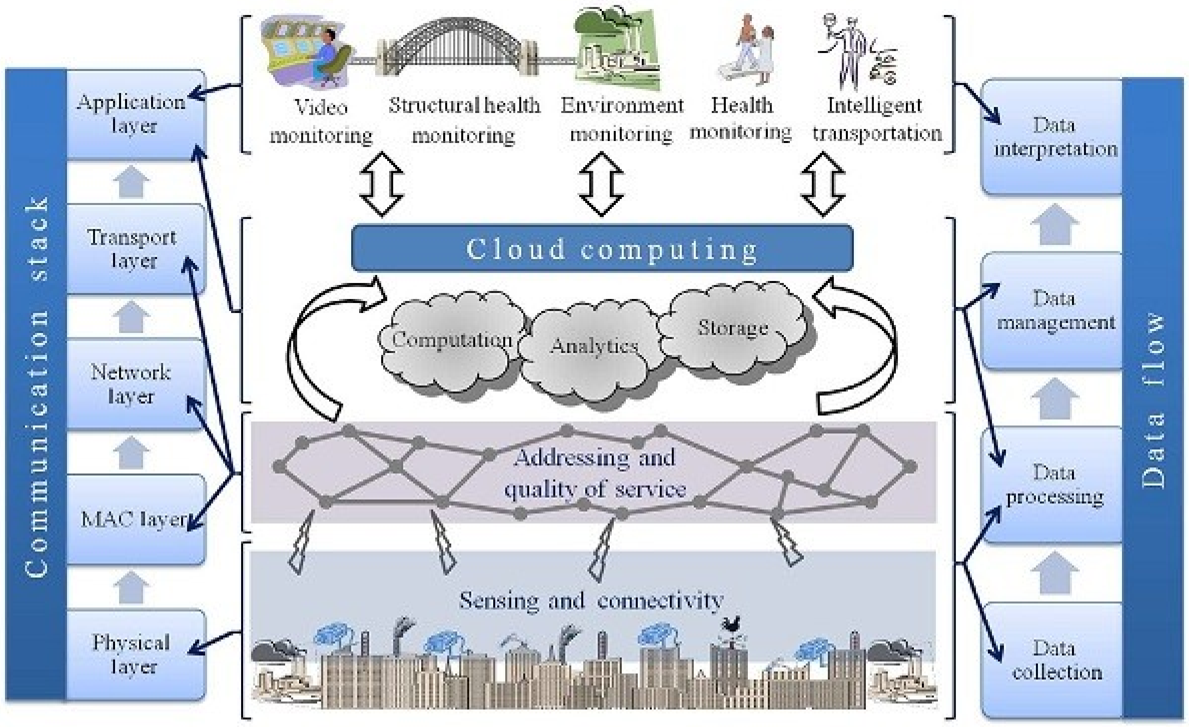


Figure 2.2: IOT ecosystem with different point of view[3]

- **Quality of Service (QoS) Mechanism:** For a different IOT application the QoS will be different like in fire alarm QoS should be high because cause of the failure of the system may damage someones life and other hand application like IP camera streaming required not much QoS in that if you lose some of the packet in return will not cause big impact on the quality of the system. For different applications there are different protocols in that different QoS can be defined for the particular application. Like in the MQTT one can define two QoS modes which we will see in further in this report.

2.2 Different models for IOT system

Here are some in general technical communication models which are used by the IOT devices. This can also be used to describe smart objects. The discussion below presents this framework and explains key characteristics of each model in the framework.

2.3 Device-to-Device Communications

The device-to-device communication model represents two or more devices that directly connect and communicate between one another, rather than an intermediary application server.

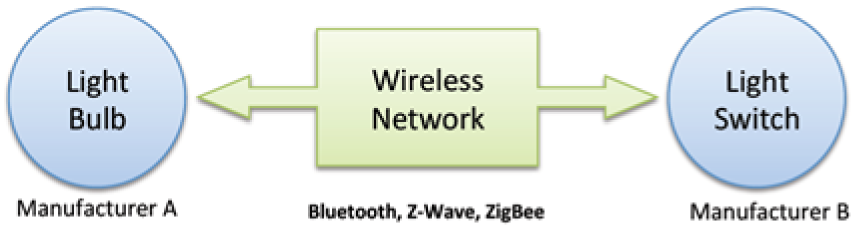


Figure 2.3: Example of device-to-device communication model[3]

This model allows devices that adhere to a particular communication protocol to communicate and exchange information, which typically use small data packets of information to communicate between devices with low data rate requirements. Residential IOT devices can have this type of communication models. Development effort is less in device-to-device communication model need to invest only in implement specific data formats rather than open approaches that enable use of standard data formats.

2.3.1 Device-to-Cloud Communications

In a device-to-cloud communication model, the device is directly connected to the cloud service and all information is exchange by cloud services. For this Device has to be connected to the IP network by using the any of the communication technology as is shown in Figure 2.

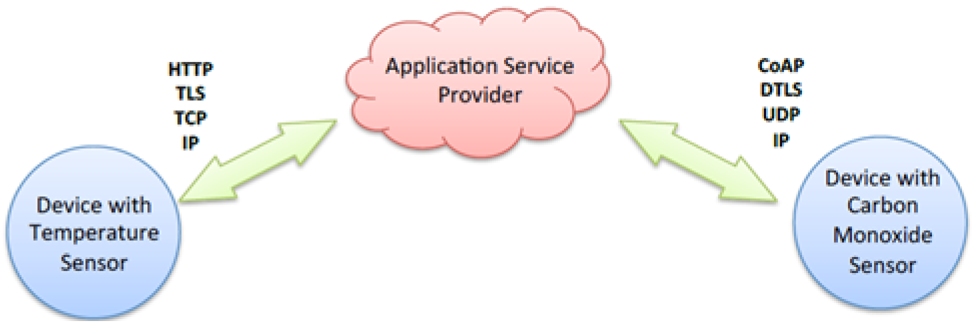

HTTP
TLS
TCP
IP
Application Service Provider
CoAP
DTLS
UDP
IP
Device with Temperature Sensor
Device with Carbon Monoxide Sensor

Figure 2.4: Device-to-cloud communication model diagram.

Device-to-cloud communication model is adopted by some popular consumer of IOT devices like the Nest Labs SmartTV and the Samsung Learning Thermostat. This model of communication has enabled the users to access device functionality remotely. In these cases, the device-to-cloud model adds value to the end user by extending the capabilities of the device beyond its native features.

In this communication model manufacturer and user has to use communication protocols which are specified by the cloud service provider. At the same time, users can generally have confidence that devices designed for the specific platform can be integrated.

2.3.2 Device-to-Gateway Model

In the device-to-gateway model, or more typically, device connects through a gateway service as a conduit to reach a cloud service. Here application software operate on the local gateway device, which act as an bridge between the application software and cloud service, so gateway has to provide security and data or protocol translation. The model is shown in Figure 3.

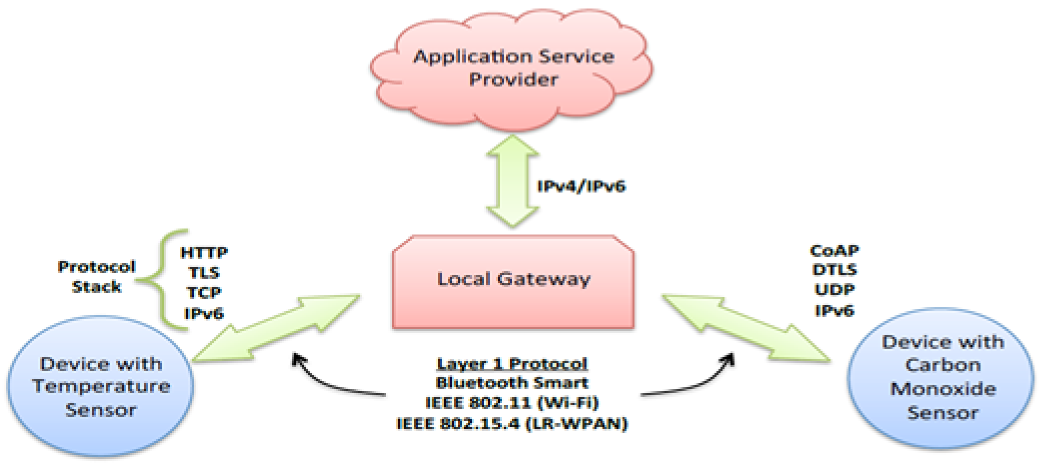


Figure 2.5: Device-to-gateway communication model diagram.

In many cases, the local gateway device is a smartphone running an app to communicate with a device and relay data to a cloud service. This is often the model employed with popular consumer items like personal fitness trackers. These devices do not have the native ability to connect directly to a cloud service, so they frequently rely on smartphone app software to serve as an intermediate gateway to connect the fitness device to the cloud.

In other words, this communications model is frequently used to integrate new smart devices into a legacy system with devices that are not natively interoperable with them. A downside of this approach is that the necessary development of the application-layer gateway software and system adds complexity and cost to the overall system.[3]

In the future, better generic bridges will come to infrastructure complexity and lower cost for the users and manufacturers. Such bridges are more likely to exist if IOT device designs make use of generic IP protocols and not required by application-layer that translate one application-layer protocol to another one.

2.4 Networking protocols for Internet of Things in Smart Home

This section briefly describes the most popular application protocols used in IOT communication, namely Hypertext Transfer Protocol-2, Constrained Application Protocol (CoAP) and Message Queue Telemetry Transport (MQTT).

2.4.1 Hypertext Transfer Protocol

HTTP protocol is based on the client and server architecture of the communication model. Where client sends the request to the server for the specific resource or the service. In HTTP first both client and server has to establish a connection after that they can communicate or transfer the requested data.

HTTP is the application layer protocol which assumes that underlying protocols are reliable. Because of this HTTP is often used with the Transmission Control Protocol (TCP). But in some rare cases HTTP can use User Datagram Protocol (UDP) as transport layer protocol.[3]

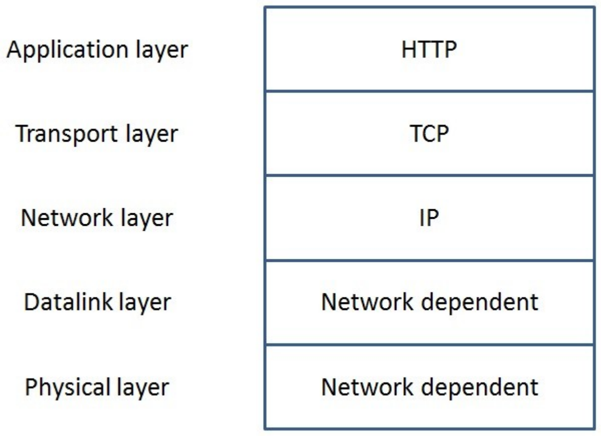


Figure 2.6: HTTP Stack.[2]

In the HTTP resources are located and identified on the network by Uniform Resource Identifier (URI) or Uniform Resource Locator (URL).

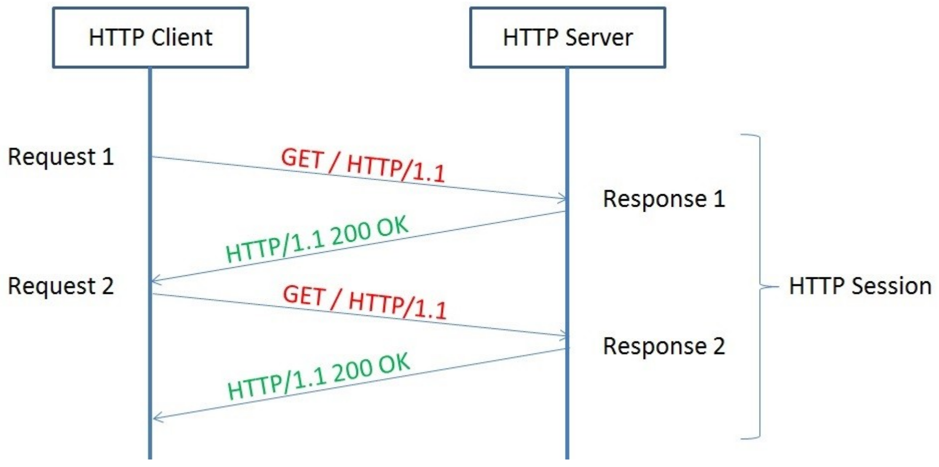
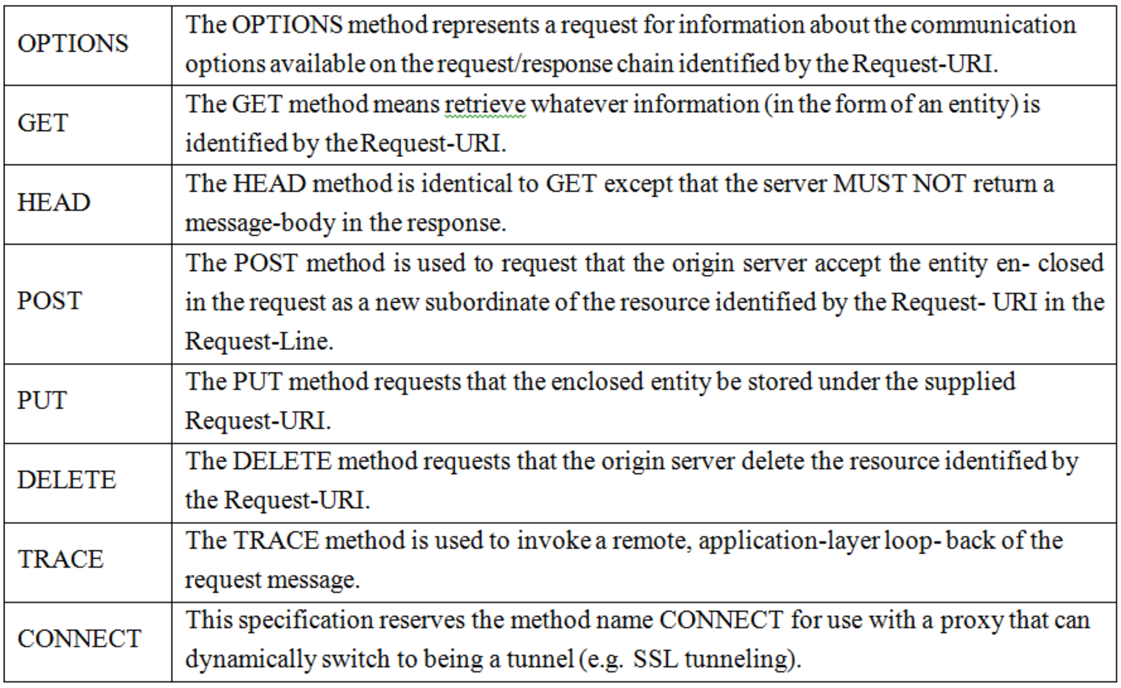


Figure 2.7: HTTP Stream.

In the HTTP protocol client will initiate request for the particular service or resource on specific port of server. HTTP server starts polling on that port for the client request. As and when client request came server respond with the acknowledgement message. Current version of HTTP/1.1 reuses the TCP connection several times to send and receive multiple HTTP request/response instead of creating a new TCP connection for every single request/response pair.

Below is the HTTP defined methods for the operation to carried out on resource.



OPTIONS	The OPTIONS method represents a request for information about the communication options available on the request/response chain identified by the Request-URI.
GET	The GET method means retrieve whatever information (in the form of an entity) is identified by the Request-URI.
HEAD	The HEAD method is identical to GET except that the server MUST NOT return a message-body in the response.
POST	The POST method is used to request that the origin server accept the entity en- closed in the request as a new subordinate of the resource identified by the Request- URI in the Request-Line.
PUT	The PUT method requests that the enclosed entity be stored under the supplied Request-URI.
DELETE	The DELETE method requests that the origin server delete the resource identified by the Request-URI.
TRACE	The TRACE method is used to invoke a remote, application-layer loop- back of the request message.
CONNECT	This specification reserves the method name CONNECT for use with a proxy that can dynamically switch to being a tunnel (e.g. SSL tunneling).

Figure 2.8: HTTP Methods.

2.4.2 HTTP-2

So whats HTTP-2 set out to do then? Where are the boundaries for what the HTTP bis group set out to do? They were actually quite strict and put quite a few restraints on the teams ability to innovate.

1. It has to maintain HTTP paradigms. It is still a protocol where the client sends requests to the server over TCP.
2. http:// and https:// URLs cannot be changed. There can be no new scheme for this. The amount of content using such URLs is too big to expect them to change.
3. HTTP1 servers and clients will be around for decades, we need to be able to proxy them to http2 servers.

4. Subsequently, proxies must be able to map http2 features to HTTP 1.1 clients 1:1.

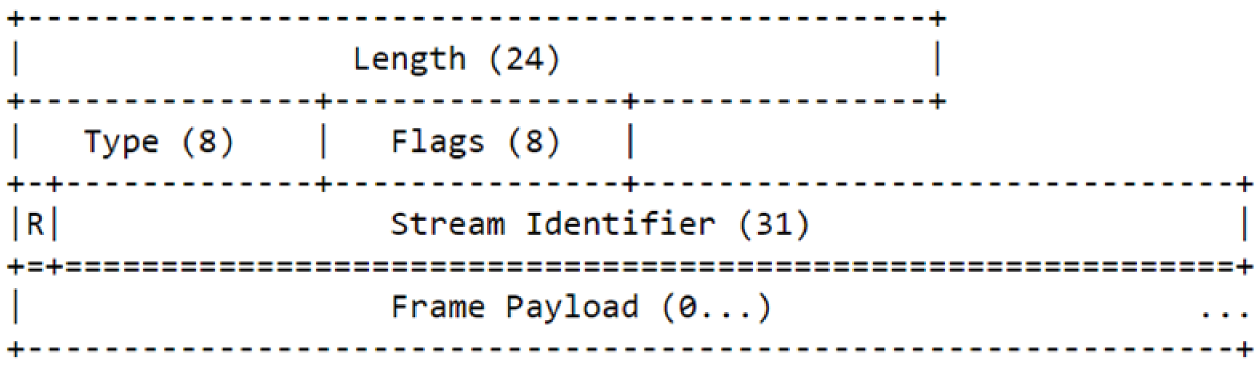
5. Remove or reduce optional parts from the protocol. This wasn't really a requirement but more a mantra coming over from SPDY and the Google team. By making sure everything is mandatory there's no way you cannot implement anything now and fall into a trap later on.

6. No more minor version. It was decided that clients and servers are either compatible with http2 or they are not. If there comes a need to extend the protocol or modify things, then http3 will be born. There are no more minor versions in http2.[2]

Below is the some of Major features of the HTTP-2.

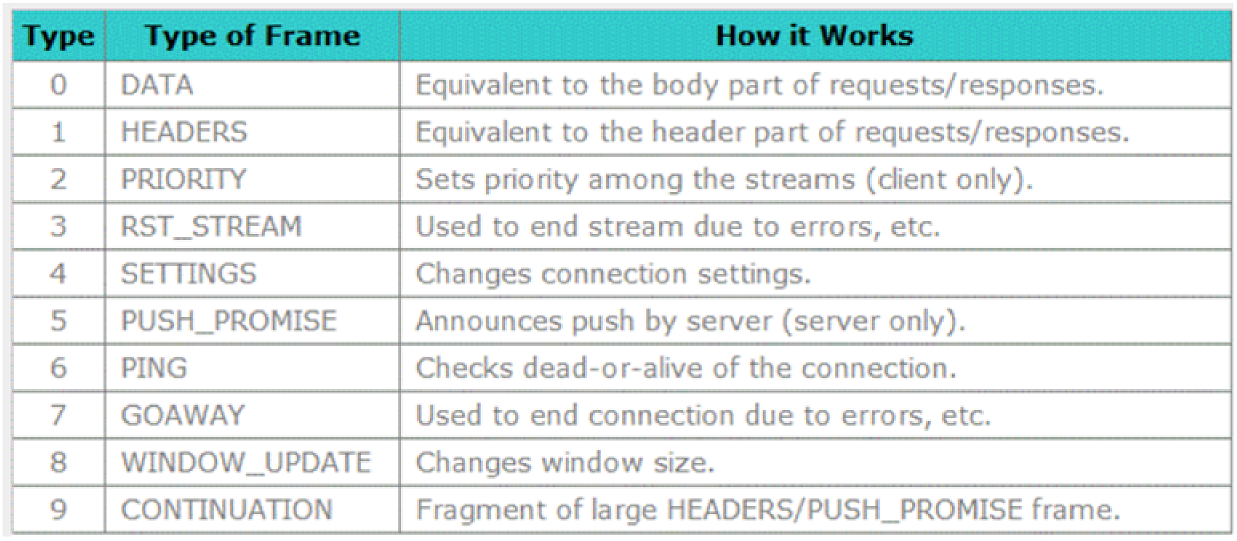
- **Binary Protocol:**
  HTTP-2 is a binary protocol which means all the text and ASCII values in the HTTP 1.1 will be converted in to the equivalent binary format. Advantage of using binary protocol is easy to frame and compress. Figuring out the start and stop of the frame is much difficult in the actual text based protocol. By using the binary protocol we can avoid the white space and other things which help us in the compression.

  Also, it makes it much easier to separate the actual protocol parts from the framing which in HTTP-1 is confusingly intermixed. The facts that the protocol features compression and often will run over TLS also diminish the value of text since you wont see text over the wire anyway.



```
+-----------------------------------------------+
|                 Length (24)                   |
+---------------+---------------+---------------+
|   Type (8)    |   Flags (8)   |
+-+-------------+---------------+-------------------------------+
|R|                 Stream Identifier (31)                      |
+=+=============================================================+
|                   Frame Payload (0...)                      ...
+---------------------------------------------------------------+
```

Figure 2.9: HTTP-2 Frames Format[1]



Type	Type of Frame	How it Works
0	DATA	Equivalent to the body part of requests/responses.
1	HEADERS	Equivalent to the header part of requests/responses.
2	PRIORITY	Sets priority among the streams (client only).
3	RST_STREAM	Used to end stream due to errors, etc.
4	SETTINGS	Changes connection settings.
5	PUSH_PROMISE	Announces push by server (server only).
6	PING	Checks dead-or-alive of the connection.
7	GOAWAY	Used to end connection due to errors, etc.
8	WINDOW_UPDATE	Changes window size.
9	CONTINUATION	Fragment of large HEADERS/PUSH_PROMISE frame.

Figure 2.10: HTTP-2 Frame Types[2]

- **Multiplexed Streams:**
  Each frame associate with the one stream id, it is the logical association between the client and server within the frame. It will help to maintain independent bidirectional sequence of frame. A single HTTP-2 connection can have multiple open streams for different application within the same socket, This feature was not there in the conventional HTTP. This future of the HTTP-2 helps to handle multiple streams on the single connection.

Recipients process frames in the order they are received. Multiplexing the streams means that packages from many streams are mixed over the same connection. Two (or more) individual trains of data are made into a single one and then split up again on the other side. In HTTP-2 we will see tens and hundreds of simultaneous streams. The cost of creating a new one is very low.

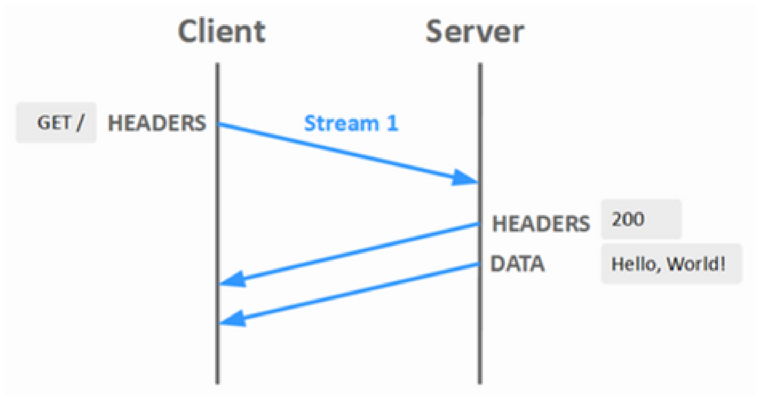


Figure 2.11: Simplest Request Exchange Frames[2]

- **Header Compression:**
  HTTP protocol is the stateless protocol which specify that every client request should have enough details to serve the request. Server will not store the metadata for the past client request. HTTP-2 also follows this paradigm to remove server load to certain extend.

  This makes HTTP repetitive. When a client asks for many resources from the same server, like images from a web page, there will be a large series of requests that all look almost identical. A series of almost identical something begs for compression.

  While the number of objects per web page increases as I have mentioned earlier, the use of cookies and the size of the requests have also kept growing over time. Cookies also need to be included in all requests, mostly the same over many requests. The HTTP 1.1 request sizes have actually gotten so large over time so they sometimes even end up larger than the initial TCP window, which makes them very slow to send as they need a full round-trip to get an ACK back from the server before the full request has been sent another argument for compression.

- **Server Push:**
  This is also known as the chase push. Idea behind this is client can ask for the particular resource and server has the prior knowledge that client will most likely do request for the particular resources. So server will send those resources without waiting for the client request. Client can chase those information or if not required client can abandon those data or information.

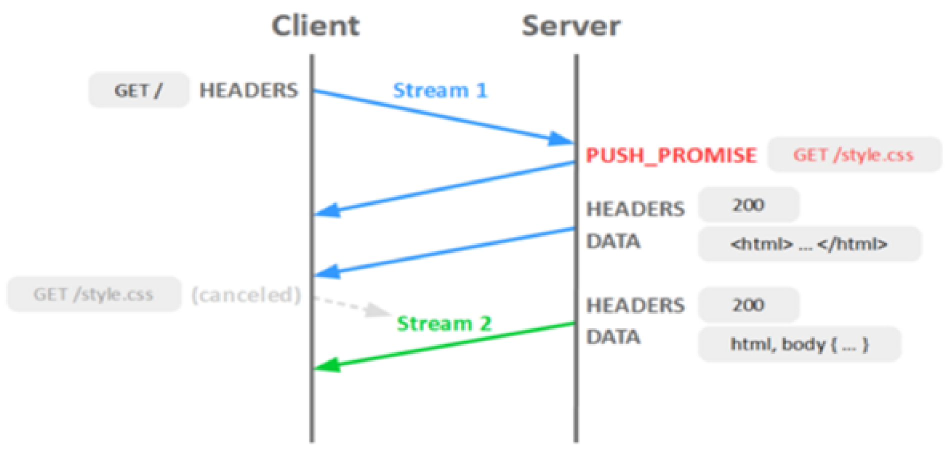


Figure 2.12: HTTP2 Server Push Request[2]

2.4.3 Constrained Application Protocol (CoAP)

COAP is easy protocol implementation for communication between embedded devices. There is packet format that can support both reliability and no reliable applications.

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 [**?**]:

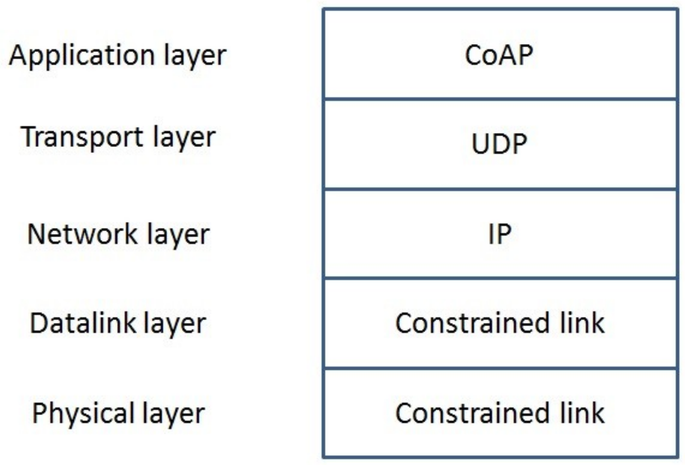


Figure 2.13: CoAP Protocol Stack

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 compared to running HTTP in that constraint nodes.
COAP provides both reliability and non-reliability support that allows COAP to use in both kind of use case or applications.

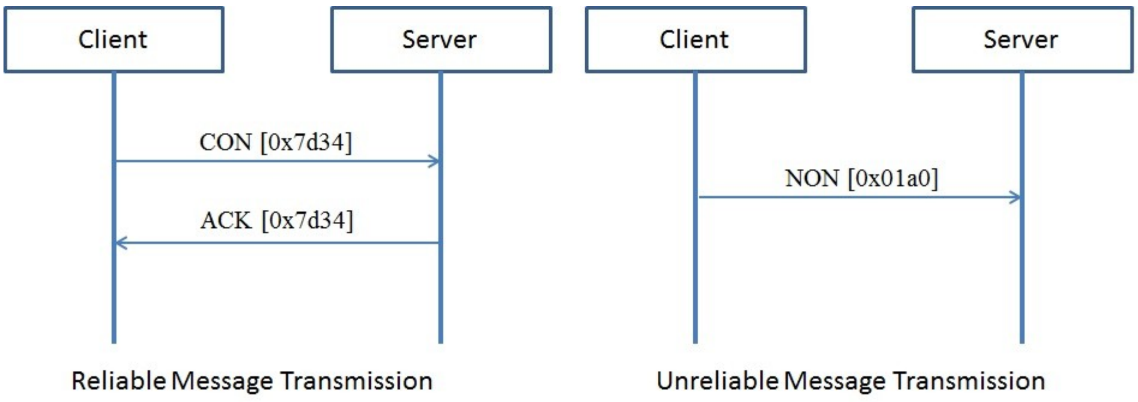
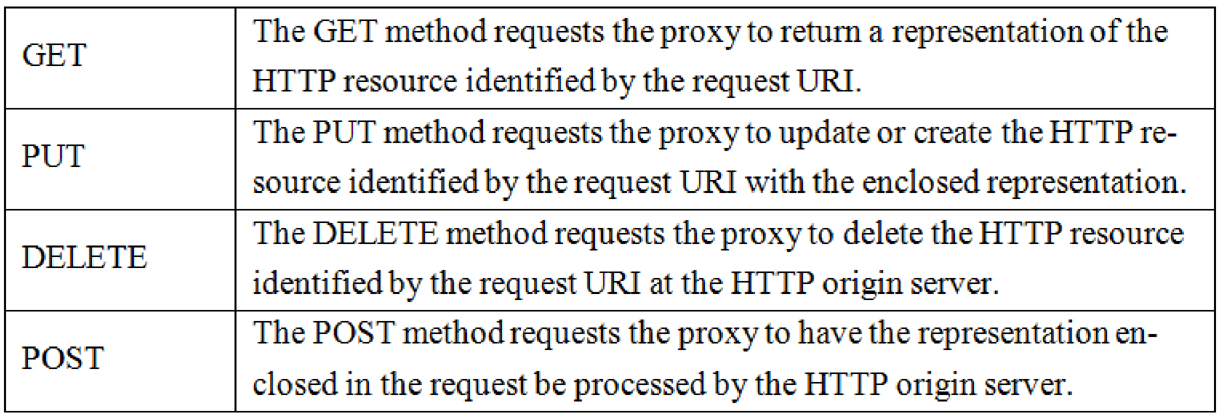

Client
Server
CON [0x7d34]
ACK [0x7d34]
Reliable Message Transmission
Client
Server
NON [0x01a0]
Unreliable Message Transmission

Figure 2.14: CoAP Messages

CoAP defines the following methods: GET, PUT, DELETE and POST.



GET	The GET method requests the proxy to return a representation of the HTTP resource identified by the request URI.
PUT	The PUT method requests the proxy to update or create the HTTP re-source identified by the request URI with the enclosed representation.
DELETE	The DELETE method requests the proxy to delete the HTTP resource identified by the request URI at the HTTP origin server.
POST	The POST method requests the proxy to have the representation en-closed in the request be processed by the HTTP origin server.

Figure 2.15: CoAP Methods

As described in above figure above message can be in the unreliable mode that will be marked as NON (Non- conformable). In this receiver server will not acknowledge the received message but still it has message ID for the duplication detection. If received message not able to served then server sends a reset message.

CoAP message are encoded in the binary format same as the HTTP-2 so the packet will be smaller compared to other protocol below figure shows message frame format of the CoAP protocol[9].

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver| T |  TKL  |      Code     |          Message ID           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Token (if any, TKL bytes) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Options (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 1 1 1 1 1 1|    Payload (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2.16: CoAP Message Frame

CoAP has one most important feature called observer functionality in that client send one GET request with the observer tag and server will push the resource data periodically to the client as in the asynchronous mode at time instant. In that whenever resource changes its state it will notify to the client about the new state of the resource. This functionality is same as the server push in the HTTP-2 protocol. Following fig. shows the observer functionality of the CoAP protocol[4]:

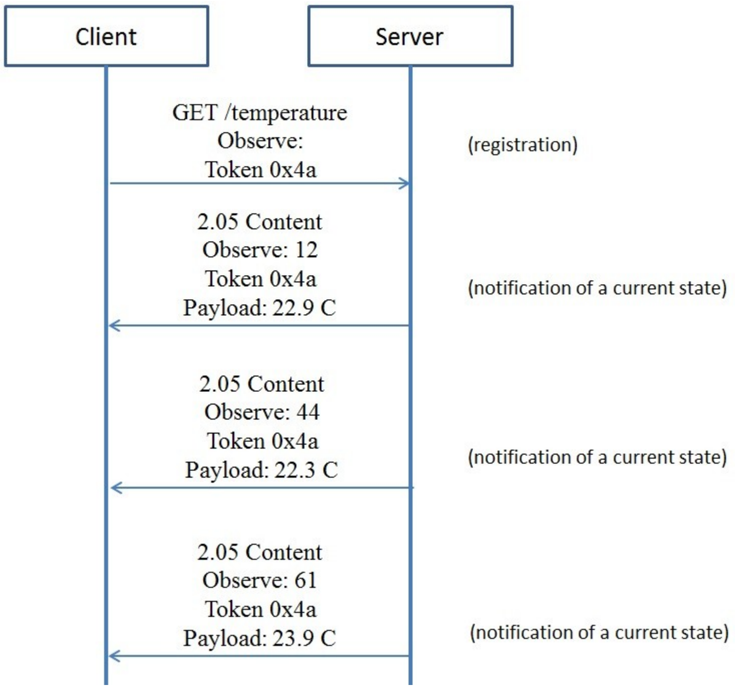


Figure 2.17: Observer Functionality of CoAP

2.4.4 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 subscribers 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.[9]

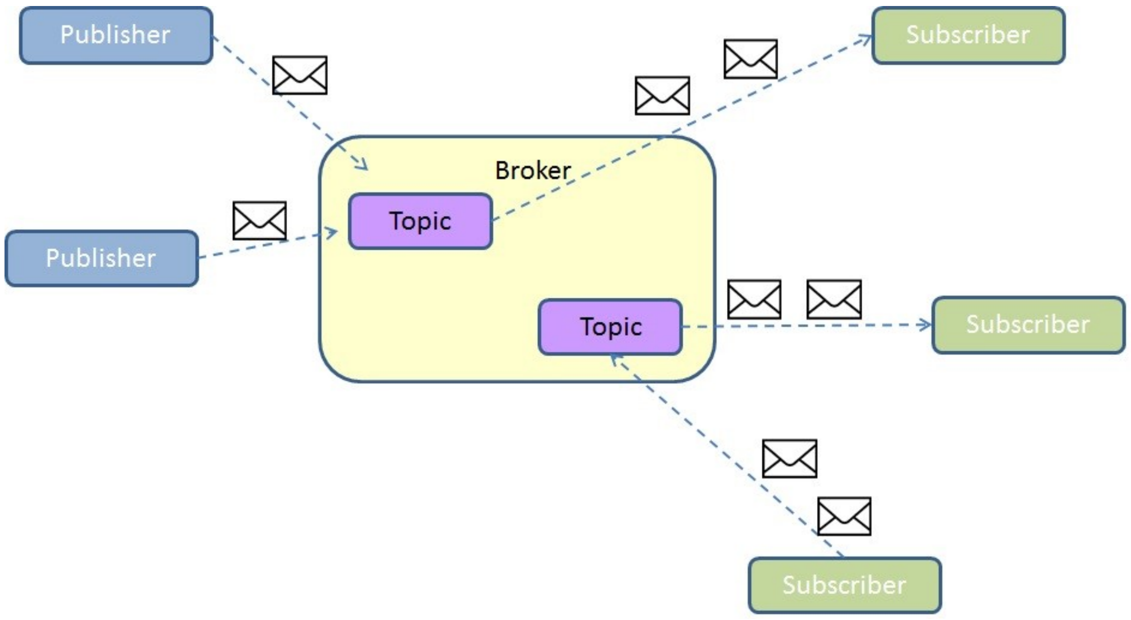


Figure 2.18: MQTT Architecture[**?**]

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 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.

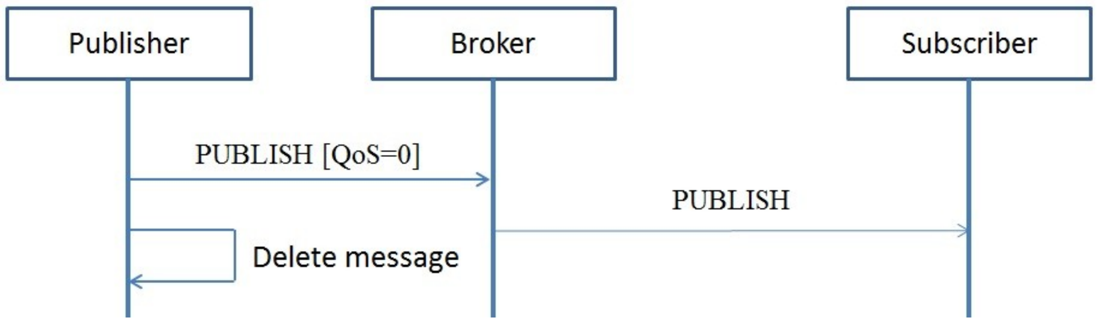


Figure 2.19: MQTT QoS-0

In QoS 0 scheme (figure 17), the message is delivered according to the best efforts of the underlying network. No response is sent back by the receiver/subscriber and no retry is performed by the sender/publisher. The message arrives at the receiver/subscriber either once or not at all.

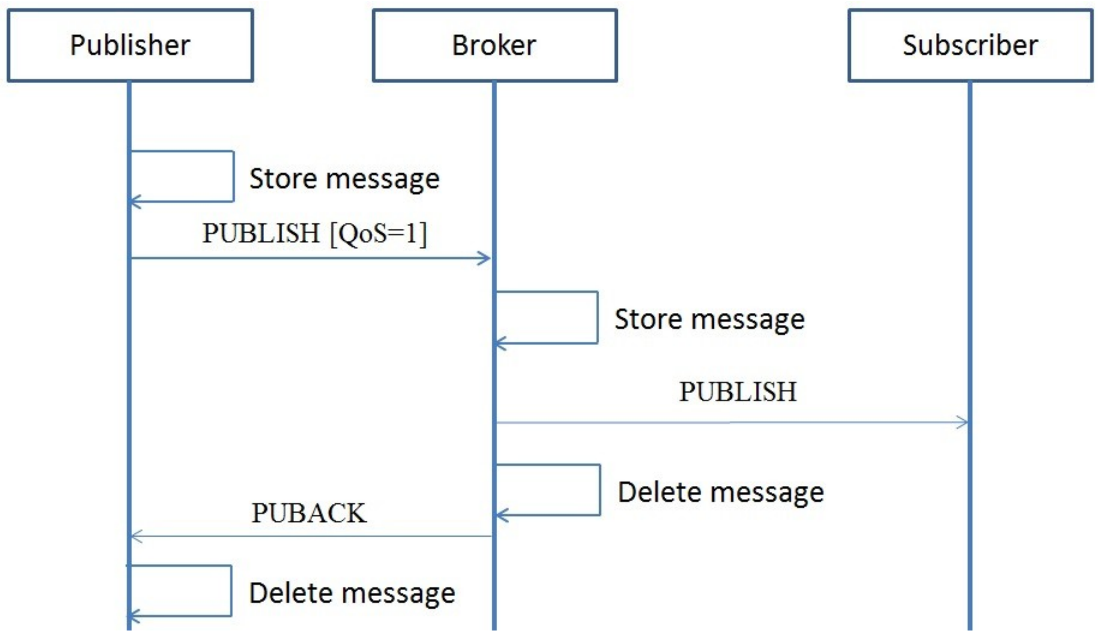


Figure 2.20: MQTT QoS-1

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 un-subscribe for particular topic by giving topic name, client id and with proper un-subscribe frame format. So once broker receives this frame broker understands that client or subscriber wants to un-subscribe 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.[9]

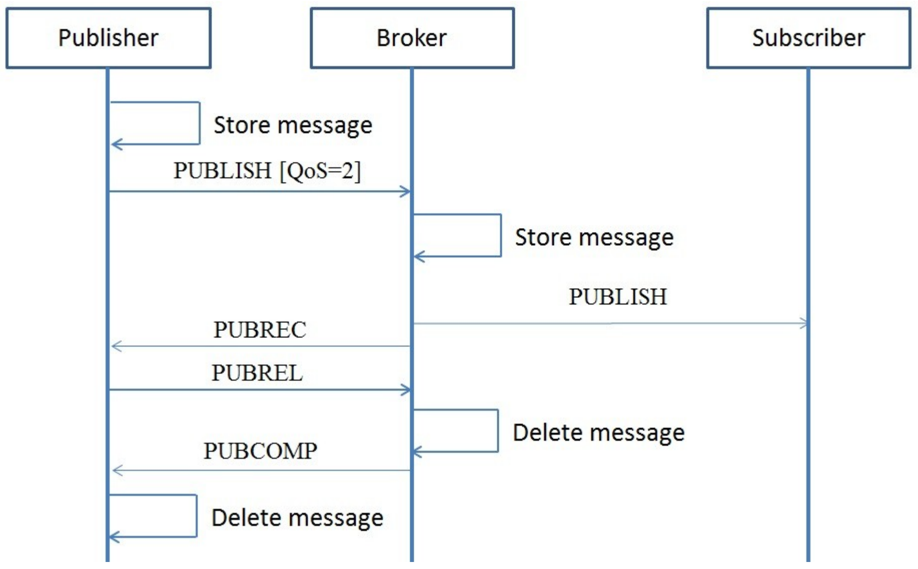


Figure 2.21: MQTT QoS-2

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.

- **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.

2.4.5 Comparison Between Protocols

The purpose of this literature survey is to analyze the comparison of above mentioned IOT protocols, so one can choose best kind of protocol for their application. We saw that some of the above protocols has similarities such as send the sensor node data to the cloud and high network constraint. Here CoAP, MQTT and HTTP2 been compared in terms of protocol architecture, performance of the protocol for different types of application and cost efficiency.

In [6], the publisher saw qualitative comparison between the CoAP and MQTT. Paper tells us that MQTT is the better option in which system requirement will be like different kind of the QoS at run time and stored messages. MQTT will take advantage in terms of the security for the multicast messages. In other application CoAP takes advantage over MQTT in terms of bandwidth requirement and round trip time.

MQTT is more efficient for the system requirement like message exchange happen frequently and CoAP has the fragmentation ability so large message can be fragmented into the smaller chunks and send efficiently onto the network.

In [7] CoAP and HTTP has been compared term of total cost. The comparison tells us that while performing same task with the both the protocols like humidity and temperature sensing CoAP perform well over the HTTP. Author took two different communication methods for comparison. one push method client device will sleeps most of the time when its awake it will communicate with AP and send the sensed data.

Second one is pull mode in that device always waiting for the request for the data and respond as an when request arrived with the sensed data. For the push method CoAP consumption rate 2 times less With respect two HTTP and for the polling CoAP consumption rate 6 times lower than the HTTP. In terms of data CoAP based system will generate 3 times less data then HTTP based system.

2.4.6 Chapter conclusion

In this chapter we saw some of the aspects of the smart and modern home paradigm. Opening with the definition of the smart home and followed by the different protocols which are widely used in the smart home system. We saw the different kind of models for the IOT system like device to device communication, device to cloud communication and device to getaway communication. Than detailed architecture of the mainly three IOT protocols CoAP, MQTT and HTTP-2 followed by protocols being compared with each other and highlighted for specific kind of application.

Chapter 3

Test Results

This chapter describes if system developed by three different protocols (MQTT, CoAP, HTTP) then what could be the behavior of the system with respect to required bandwidth and volume of generated data. By using company's test setup here describe how one IoT system behave differently for different protocol. Here in this chapter uplink channel is defined for the server which is basically for monitoring entire system and downlink channel for the IoT device (end node). From this exercise determined that which component is inefficient with respect to the IoT scenario and can be improve over the period of time.

In this chapter most of the results taken by using companys test setup which has capability to test some systems with the different architecture and different protocols.

3.1 Defined Traffic Pattern

Here we have generated the defined traffic pattern which is mostly used in the Internet of things based systems. Setup is like for MQTT one gateway will send traffic at defined period of time.

To/from IP address 89.18.105.52 this is a device IP address which is receive the sensor data from the Telia server.

3.2 Periodic Traffic Pattern

Pattern-1:

In the pattern-1 Telia server will send the sensor data at every 60 seconds gateway sends 74 bytes of application data to the device this pattern-1 corresponds to the PINGREQ and PINGRESP packets of the MQTT protocol mentioned in above Chapter. Following figure is describing more about the system.

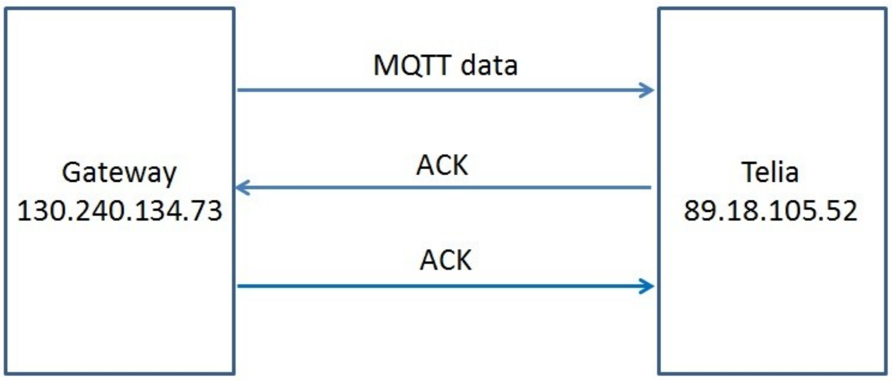


Figure 3.1: Periodic Traffic Pattern MQTT-1

Pattern-2:

In pattern 2, every 240 seconds the gateway sends an application layer packet of 970 bytes, followed by a transport layer ACK of 66 bytes from 89.18.105.52. The process is detailed in the following figure:

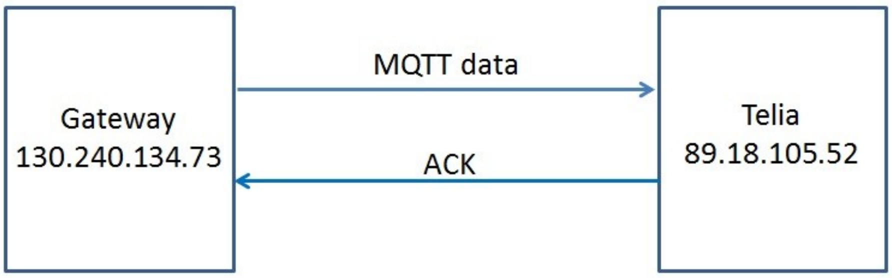
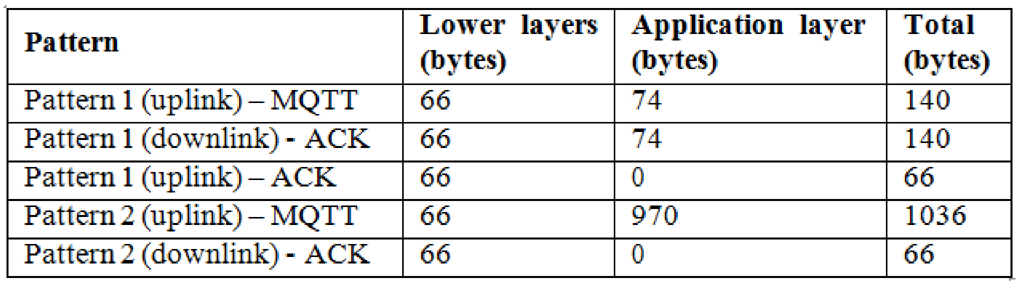


Figure 3.2: Periodic Traffic Pattern MQTT-2

The following table summarizes the data generated by pattern 1 and 2 in bytes:



Pattern	**Lower layers (bytes)**	**Application layer (bytes)**	**Total (bytes)**
Pattern 1 (uplink) – MQTT	66	74	140
Pattern 1 (downlink) - ACK	66	74	140
Pattern 1 (uplink) – ACK	66	0	66
Pattern 2 (uplink) – MQTT	66	970	1036
Pattern 2 (downlink) - ACK	66	0	66

Figure 3.3: Summary for MQTT

Pattern-3:

Here the pattern-3 shows when system uses the encryption like Diffie-Hellman Key Exchange Algorithm. Device will receive the encrypted packet form server every 60 second. Server sends the two encrypted packet using SSHv2 each of 36 bytes followed by 2 ACK packet. Below figure shows such system model.

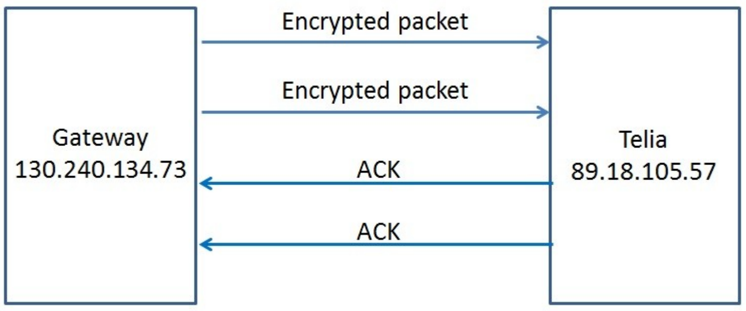
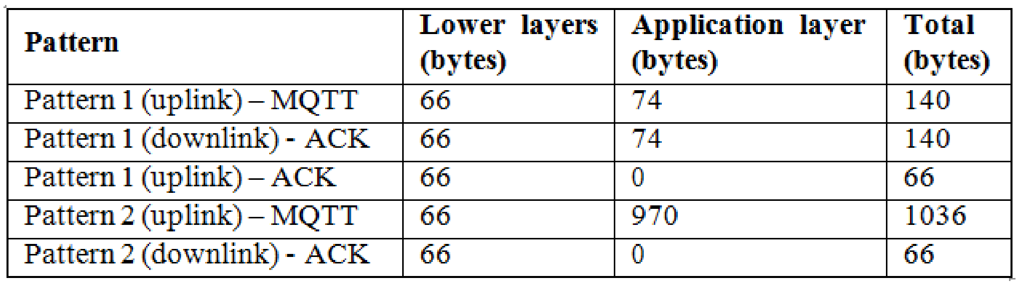


Figure 3.4: Encrypted Traffic

The following table summarizes the data generated by pattern 3 in bytes:



Pattern	**Lower layers (bytes)**	**Application layer (bytes)**	**Total (bytes)**
Pattern 1 (uplink) – MQTT	66	74	140
Pattern 1 (downlink) - ACK	66	74	140
Pattern 1 (uplink) – ACK	66	0	66
Pattern 2 (uplink) – MQTT	66	970	1036
Pattern 2 (downlink) - ACK	66	0	66

Figure 3.5: Data Generated from Pattern-3

Pattern 4:

Below system is based on the HTTP protocol, this pattern describe the behavior of the HTTP protocol in this client first establish the connection and client is responding with the 200 OK status code and ACK packet for the TCP connection. This whole process happens at every 240 seconds. Following figure shows the packet flow.

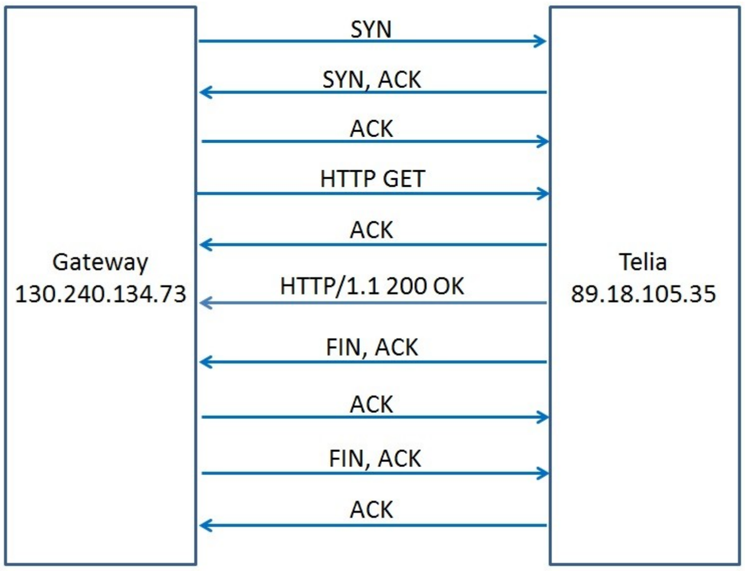


Figure 3.6: Packet Flow for Pattern-4

The following table summarizes the data generated by pattern 4 in bytes:

Pattern	**Lower layers**	**Application layer**	**Total**
Pattern 4 (uplink) – SYN	74	0	74
Pattern 4 (downlink) - SYN, ACK	74	0	74
Pattern 4 (uplink) – ACK	66	0	66
Pattern 4 (uplink) - HTTP GET	66	82	148
Pattern 4 (downlink) – ACK	66	0	66
Pattern 4 (downlink) - HTTP 200 OK	66	226	292
Pattern 4 (downlink) - FIN, ACK	66	0	66
Pattern 4 (uplink) – ACK	66	0	66
Pattern 4 (uplink) - FIN, ACK	66	0	66
Pattern 4 (downlink) – ACK	66	0	66

Figure 3.7: Data Generated from Pattern-4

3.3 Video Streaming

Pattern 5:

This pattern is about based on the cloud based video server is connected to the client. The exchange starts with the TCP handshake followed by the data frames. This pattern run over the every 60 seconds, whole process is detailed in the following figure.

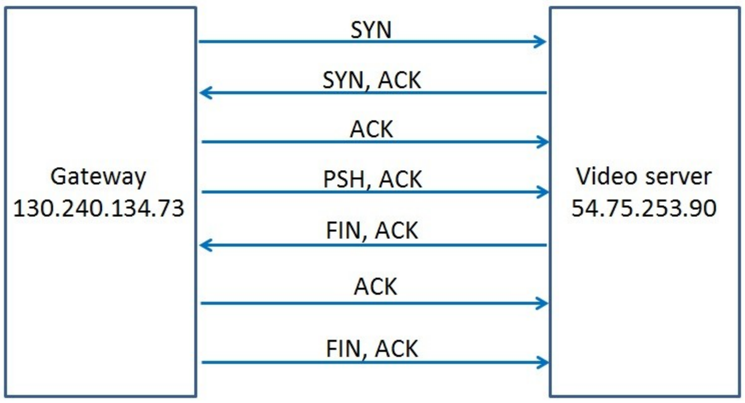


Figure 3.8: Packet Flow for Pattern-5

Pattern 6:

In this pattern gateway exchanges TCP Keep-Alive packets with the video server. The gateway sends TCP Keep-Alive packet to the server; the server responses by sending TCP Keep-Alive ACK packet. The process happens every 30 seconds. It is detailed in the following figure:

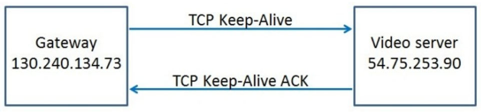
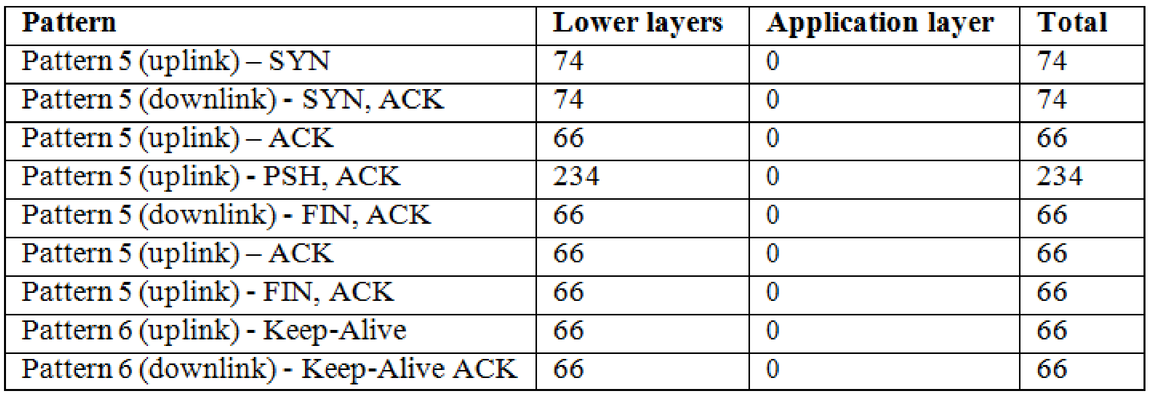


Figure 3.9: Packet Flow for Pattern-6

The following table 10 summarizes the data generated by pattern 5 and 6 in bytes:



Pattern	**Lower layers**	**Application layer**	**Total**
Pattern 5 (uplink) – SYN	74	0	74
Pattern 5 (downlink) - SYN, ACK	74	0	74
Pattern 5 (uplink) – ACK	66	0	66
Pattern 5 (uplink) - PSH, ACK	234	0	234
Pattern 5 (downlink) - FIN, ACK	66	0	66
Pattern 5 (uplink) – ACK	66	0	66
Pattern 5 (uplink) - FIN, ACK	66	0	66
Pattern 6 (uplink) - Keep-Alive	66	0	66
Pattern 6 (downlink) - Keep-Alive ACK	66	0	66

Figure 3.10: Data Generated from Pattern

Chapter 4

Implementation

4.1 Provisioning of an accessory in the home network

Provisioning of an accessory in the home network is called as an Onboarding. Onboarding service provides a common and simple way for new device to be brought onto the Wi-Fi network. This is especially useful for devices that have a limited user interface, like a SmartPlug. The current onboarding mechanism leverages Wi-Fi only, though the system can evolve to leverage additional hardware (like BTLE) as they become more relevant in these classes of devices.

Below is the flow diagram of the whole onboarding process of the any new device. As shown in figure when device boots up for the first time it does not have information about which AP it should connect. In that case user has to give AP credentials to the device so that it can connect to the specific AP. When user gives the information about the AP it should also verify the authenticity of the device.
When device boots up first time it has to be in the softAP mode so that nearby devices can discover that device and connect to it. At the time of the discovery the device also publishes its supporting profile like some device only support the lighting profile so other device can understand and act accordingly. Device can have many profiles like sensor,

thermostat, window, Door etc.

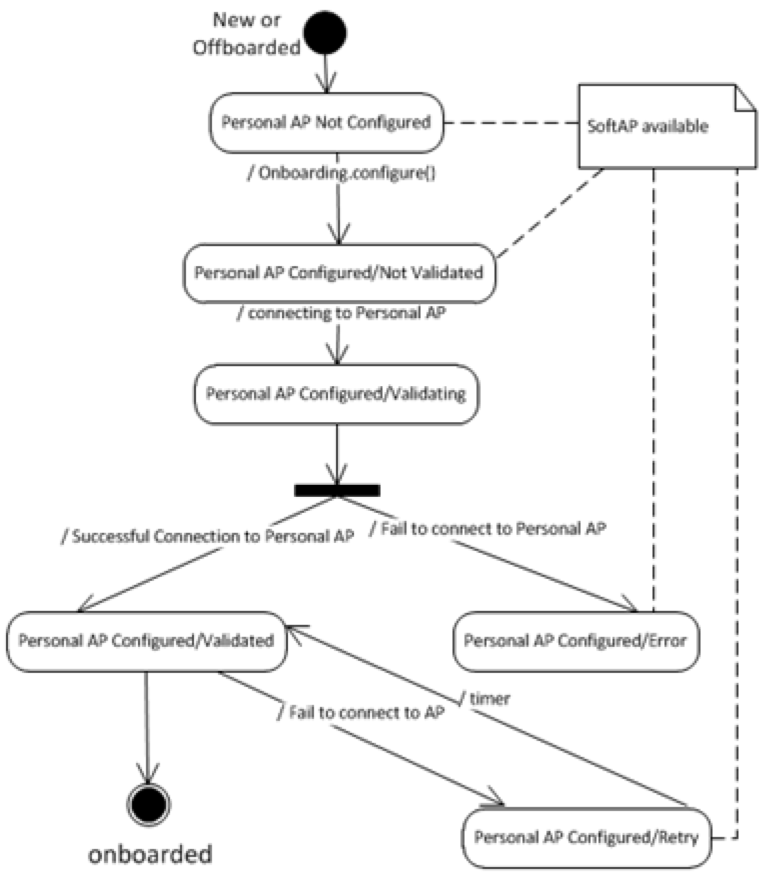

New or Offboarded
Personal AP Not Configured
SoftAP available
/ Onboarding.configure()
Personal AP Configured/Not Validated
/ connecting to Personal AP
Personal AP Configured/Validating
/ Successful Connection to Personal AP
/ Fail to connect to Personal AP
Personal AP Configured/Validated
Personal AP Configured/Error
/ timer
/ Fail to connect to AP
Personal AP Configured/Retry
onboarded

Figure 4.1: On-boarding Process[5]

I have implemented APIs that can initialize those profiles and other device can leverage devices characteristics and services after onboarding process. APIs give the simplicity and modularity to the application developer. Developer can use those APIs to make any of the above device profile and can access its functionality.

I have created the snippet applications to demonstrate the usage and functionality of those APIs to the developers. Also created the test application for the profiles, it is console based application where user can read data and write data into the profiles for the testing the functionality of profile.

4.2 Specific Scan Feature

- **Problem statement:**In the onboarding process when user does not give the information about the security mode in which the AP is then application itself should be able to find the security type of the given AP by scanning the surrounding with the specific AP name.

- **Difficulties:**In existing implementation WiFi scan functionality has implemented as an asynchronous event so it returns all surrounded APs object. It has to be handled into the callback function but in above case we have the partial information about the AP, so we can use that information to speed up the scan process and it can be implemented as synchronous.

- **Solution:**I have created the new API for the user to expose this functionality into the application level. User has to give known information about the AP and pointer of AP structure so that in return path API can return the AP information in to that structure.

- **Lesson learned:**I have learnt how the Wifi network has established and get the information about the nearby Access point from the beacon frames. Structure of the beacon and information ware carrying the beacon frame. Parsing the frame and extract the useful information from the beacon frame.

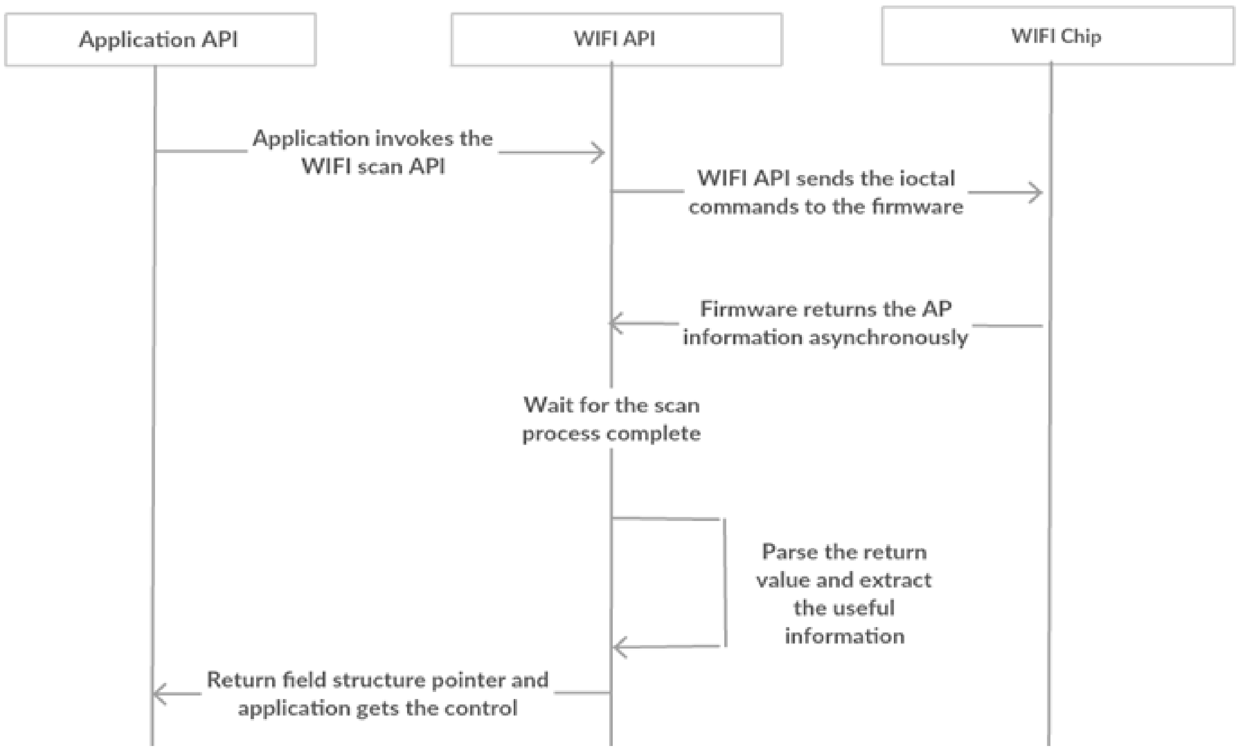

Application API
WIFI API
WIFI Chip
Application invokes the WIFI scan API
WIFI API sends the ioctal commands to the firmware
Firmware returns the AP information asynchronously
Wait for the scan process complete
Parse the return value and extract the useful information
Return field structure pointer and application gets the control

Figure 4.2: Specific Scan Process

Chapter 5

Conclusion and Future Scope

5.1 Conclusion

Internet of things helps to monitor your things,useful information remotely by connecting sensors to cloud. Technology helps to improve quality of life. The implementation of protocols like HTTP-2, COAP and MQTT are very useful for the M2M communication where one device communicates to another without human intervention. So main goals for these protocols are to reduce power consumed by constrained device, less bandwidth and less data generation where HTTP 1.1 consumes lot more power for this constrained device. It helps increasing the lifetime of constrained devices and also allows machine to machine communication without any human intervention. In the security aspects use of TLS over TCP is also important part here , because security is major concern in Internet of things. It is required that communication with the device has to be proper in authentication and consistent in integrity.

5.2 Future Scope

In the context of IOT the connectivity is a one of the pillars and connectivity relies on the protocols. By doing this project, one can learn to choose the correct protocol or set of a few protocols which is known to have the right characteristics for the application deployment, management and application support, the best implementation of each protocol can be understood. From this understanding, the designer can select the optimal implementation of each protocol for the system. Also one will be able to select the best protocol implementation for the system.

References

[1] RFC 7540, Hypertext Transfer Protocol Version 2 (HTTP/2)". IETF. May 2015. Retrieved May 14, 2015.
$https://tools.ietf.org/html/rfc7540$

[2] Thomson, M. (ed.), Belshe M. and R. Peon. "Hypertext Transfer Protocol version 2 - draft-ietf-httpbis-http2-16". ietf.org. HTTPbis Working Group. Retrieved February 11, 2015.

[3] Karen Rose, Scott Eldridge and Lyman Chapin, ISOC-IoT-Overview, Internet Society, Geneva, Switzerland, Octomber-2015.

[4] Daniel Stenberg, HTTP2 Explained, ACM SIGCOMM Computer Communication Review, Volume 44, Number 3, July 2014.

[5] Allseen Allianc, "Alljoyn tutorial" Website, Sept 2015,
$https://allseenalliance.org/$

[6] Paolo Patierno. IOT protocols landscape, Jun 2014.

[7] Niccolo De Caro, Walter Colitti, Kris Steenhaut, Giuseppe Mangino, and Gianluca Reali. Comparision of two lightweight protocols for smartphone-based sensing. In Communications and Vehicular Technology in the Benelux, 2013 IEEE 20th symposium on, 1-6. IEEE, 2013.

[8] Application protocol COAP. Coap: An application protocol for billions of tiny internet nodes. 2012.

[9] Arlen Nipper Andy Standford-Clark. Mqtt version 3.1.1, October 2014.