OULU UNIVERSITY OF
APPLIED SCIENCES

Simon Mensah

INTERNET OF THINGS:
A Review on Connectivity Gateway Protocols and Semantic Interoperability

INTERNET OF THINGS:
A Review on Connectivity Gateway Protocols and Semantic Interoperability

Simon Mensah
Master's Thesis
Autumn 2017
Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology

Author: Simon Mensah

Title of the Master's Thesis: Internet of Things. A Review on Connectivity Gateway Protocols and Semantic Interoperability
Supervisor: Kari Laitinen
Term and year of completion: Autumn 2017 Number of pages: 72

The main objective of this Master's thesis was to present a detailed overview of the most promising protocols designed for the Internet of Things (IoT) application implementation. The objective was also to serve as a comprehension for new researches and application developers to choose the best protocol for their applications deployment. A review on the existing IoT architectures, the protocol stacks, IoT gateway performance and data management with semantic interoperability of the protocols were presented to serve as a guide for developers. Also, a quick overview on the upcoming 5G cellular technology, which has been planned to have more promising technology for IoT full deployment is also presented to give an idea of what IoT will be in near future.

This thesis work was conducted mainly by a collection of relevant scientific papers and approved standards of the Internet of Things (IoT) technology. Also, players in the IoT industry were personally contacted for further real-time application implementation challenges and the constrains they face in terms of the choice of protocol and the interconnectivity or interoperability with other applications due to different protocol standards.

As long as there is no common standard for IoT protocol implementation, the result of this study will serve as a guide for IoT application developers to help them to choose the right application protocol when developing an IoT product. Again, due to the lack of common standard for IoT, interconnectivity or interoperability between devices from different vendors is a challenge for consumers, hence, the result of this thesis will help consumers to choose carefully from the vast IoT products on the market today in other to interoperate the product the buy. However, future studies on this subject could be conducted to investigate how to achieve a semantic interoperability among the application layer protocols presented in this work so that data from one vendor application can be represented in the similar format in another vendor application.

Keywords: Internet of Things, Protocol, CoAP, MQTT, HTTP, IoT gateway, 5G

PREFACE

I would like to thank my supervisor Kari Laitinen for his guidance and dedication to supervise this thesis. My sincere gratitude also goes to my family who has always stood by me and encouraged me to strive to the end. Finally, I thank Kaija Posio from ICT department for reviewing my work and also sincere thanks to all my friends.

Oulu, 13.11.2017

Simon Mensah

TABLE OF CONTENTS

ABBREVIATIONS

AMQP	Advanced Message Queuing Protocol
CDF	Cumulative Distribution Function
CoAP	Constrained Application Protocol
CONNECT	Connection request
CoRE	Constrained RESTful Environments
DTLS	Datagram Transport Layer Security
FTT	File Transmission Time
GDP	Gross Domestic Product
GIS	Geographic information system
IDC	International Data Corporation
IETF	Internet Engineering Task Force
IIoT	Industrial Internet of things
IoE	Internet of Energy
IoM	Internet of Media
IoP	Internet of People
IoS	Internet of Services
IoT	Internet of Things
IP	Internet Protocol
ITU	International Telecommunication Union
LTE	Long Term Evolution
MCPTT	Mission Critical Push-to Talk
MCS	Mission Critical Services
MIMO	Multiple Input Multiple Output
MIoT	Massive Internet of Things
MQTT	Message Queue Telemetry Transport
NACK	Negative Acknowledgement
NFV	Network Functions Virtualization
OIC	Open Interconnect Consortium
QoS	Quality of Service
REST	Representational State Transfer
RIT	Radio Interface Technologies

RTT	Round-Trip Time
SDN	Software-Defined Network
SenML	Sensor Markup Language
SMS	Short Message Services
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
SSN	Sensor Semantic Network
TCP	Transmission Control Protocol
TLS	Transport Level Security
TLV	Type-Length-Value
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URLLC	Ultra-Reliable and Low Latency Communications
VPN	Virtual Personal Network
WSN	Wireless sensor network
WWW	World Wide Web

1 INTRODUCTION

Over the past decades, an effort has been made by the information and communications technology industries to continuously increase the number of Internet enabled devices. These devices, besides the traditional computers and mobile devices, are devices that ranges from home or domestic appliances, industrial machinery and automation, healthcare, transport, energy, buildings, cities and people are been connected to the Internet. Adding more devices, which were traditionally offline to the Internet, has become possible or feasible due to the technological advancement with the hardware, software developments and the idea of network convergence known as the Internet Protocol (IP) convergence. This avalanche of many new devices and other things being connected to the Internet was known as the evolution of the Internet, which is nowadays termed as the Internet of Things (IoT).

The main idea of IoT is to connect things that are not yet connected to the Internet and to provide interconnectivity between other devices and the things to the global information and communications infrastructure. This interconnectivity of things will allow not only communication between devices and things but it will offer intelligence to the things being connected and also makes their data available to other network systems to utilize.

However, different devices from different manufacturers having different hardware platforms and networking protocols exist within the IoT, which makes it heterogeneous network of things. The interaction or interoperability with diverse devices from different manufacturers with different service platforms and networks need to be adapted to realize IoT applications. Moreover, the IoT networks could be complex due to the dynamic state of some devices and the things within the IoT. This means that some connected devices can change their states from, for example, sleeping to waking up, connected to disconnected as well as in the context of a device location and speed. The number of connected devices can change dynamically at any particular time which means that the number of devices that need to be managed will be of enormously high scale. Data collection and management from different sources is also critical to IoT applications.

1.1 The Internet of Things (IoT)

In recent times, the most widely discussed term in the wireless communications technology field of engineering is the Internet of Things, abbreviated IoT. The phrase "IoT" was first used by Kevin Ashton in 1999 (Ashton, 2009) when he was making a presentation to Procter & Gamble. In his presentation, he asserted that not only humans should generate or capture and create data but computers and other embedded devices should be able to gather their own information by sensing or interacting with their internal states or external environments. In effect, the introduction of IoT, other devices or "things" will extend the traditional Internet by making network connections more relevant and valuable than ever before and also add an entire new meaning to the information and communication technology field. In short terms, the IoTs can be described more transformational than the traditional Internet they have will have an effect on the way people live.

The IoT is a network of physical objects or "things" communicating with each other. They are embedded with electronics, sensors and actuators with computing power, software and network connectivity that enables users becoming an integral part of it. The IoT has gone through a lot of development and considerations with different definitions based on the Internet. Dr. Ovidiu Vermesan and Co. (Dr.Vermesan, et al., 2011) in their work described the term by considering the greater internet working as the Internet of Energy (IoE), Internet of Media (IoM), Internet of People (IoP) and Internet of Services (IoS). Cisco (Evans, 2012) decided to coin the term as the Internet of Everything where it was viewed as a system comprising of things, where Process, data and people together formed a "Network of Networks". In Cisco view, the IoE will connect People, process, data and the "things" together to form a network suitable and beneficial to aid in tracking "things" and also to deal with some global challenges, such as drought, climate change, sources or drinkable water, and hunger.

The IoT is fast expanding and the application areas as listed by (Asín & Gascón, 2016) include smart cities, smart water, smart metering, security and emergency, retail, logistics, industrial control, smart agriculture, smart animal faming, domestic and home automation and eHealth. However, since the application areas cover different environments and the devices involved are diverse, it makes the IoT very heterogeneous and hence challenges and barriers, such as connectivity, power management, complexity, rapid development, security and quality of service, which are always associated with wireless sensor network (WSN) standard challenges, were listed by Chase (Chase, 2013) as a development impediment of the IoT. Other challenges that Gubbi (Gubbi, Buyya, Marusic, & Palaniswami, 2013) and his colleagues noted were privacy, participatory sensing, data analytics, geographic information system (GIS) based visualization and cloud computing. Moreover, the IoT connectivity challenge also come with the architectural and protocol challenges that (Gubbi, Buyya, Marusic, & Palaniswami, 2013) considered in their work as an open challenge.

Today's industrial equipment manufacturers are confronted if not with all but most of the above-mentioned challenges when preparing products for the IoT. Therefore, for the IoT to work successfully and to meet the predicted volume of devices that are connected to the Internet by 2020, an analysis shows that it needs to be built on open, flexible hardware, software and networking platforms, which are capable of evolving and adapting. However, in this thesis work the challenge of IoT connectivity gateway protocols and their interoperability are reviewed.

1.2 IoT review

Ever since industries started to connect virtually every device and "things" from trash cans to thermostat in an event of collecting real time data, nowadays businesses have been becoming aware that the real value in the IoT is not just the data collection and processing, but it is the analyzing of the data to derive a business insight.

According to International Data Corporation's (IDC) (MacGillivray, 2016) predictions, the IoT market will reach seven billion dollars ($7,065B) globally by 2020, which will be a jump from two billion dollars ($2,715B) in 2015. Figure 1 below illustrates the IDC prediction tree of the IoT.

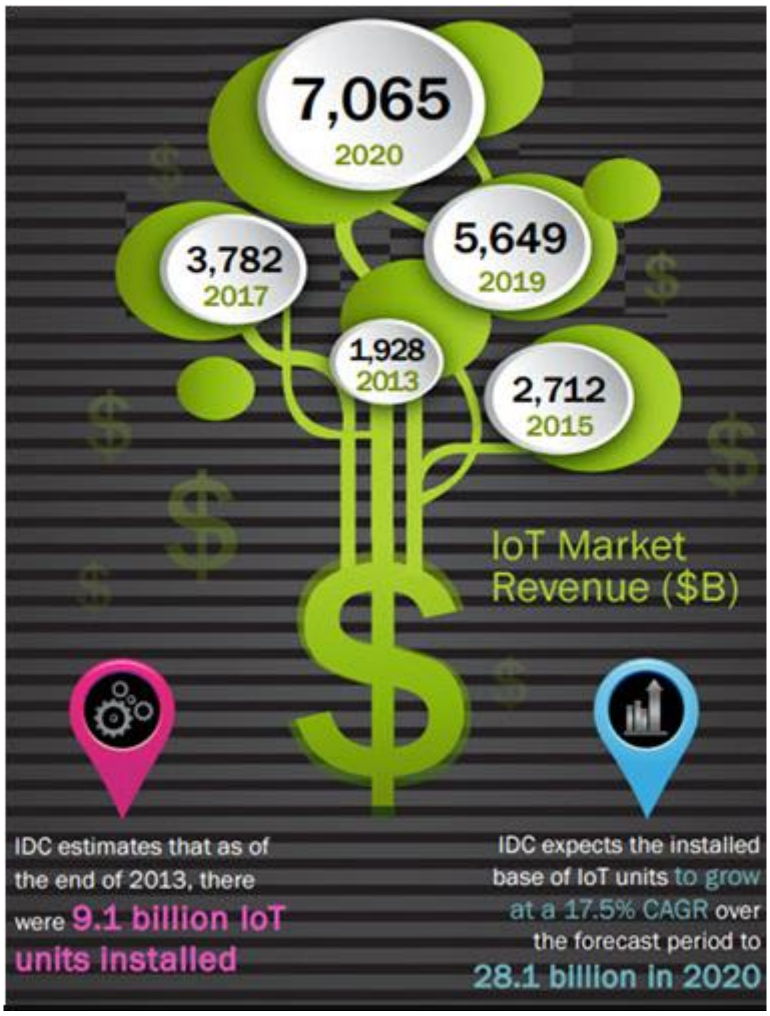


FIGURE 1. IDC IoT Market Revenue ($B) (MacGillivray, 2016)

The Gartner (Meulen, 2015) prediction in 2015 pointed out that by 2016, 6.4 billion "things" will be connected worldwide and by 2020 the number of connected devices will reach 20.8 billion. Cisco (Bradley, Reberger, Dixit, & Gupta, 2013) has also predicted that the value of the Internet of Everything through cost savings, productivity gains, new revenues and improving citizen experiences could generate $4.6 trillion globally by 2022 in the public sector. In addition, McKinsey (Ip, 2016) estimated that the size of the total IoT market in 2015 was risen to $900Million and this will grow to $3.7billion by 2020. Figure 2 illustrates an IoT potential economic impact by 2025 captured by McKinsey Global Institute analysis.

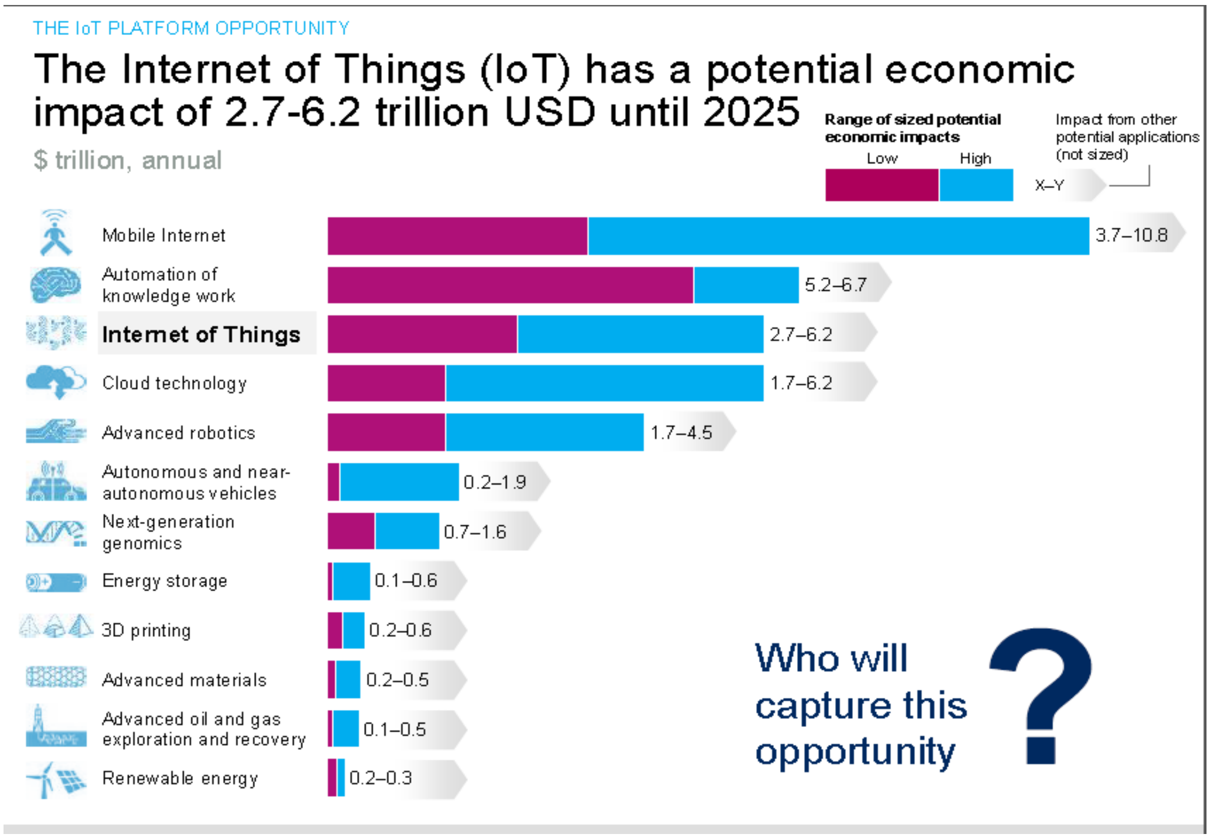
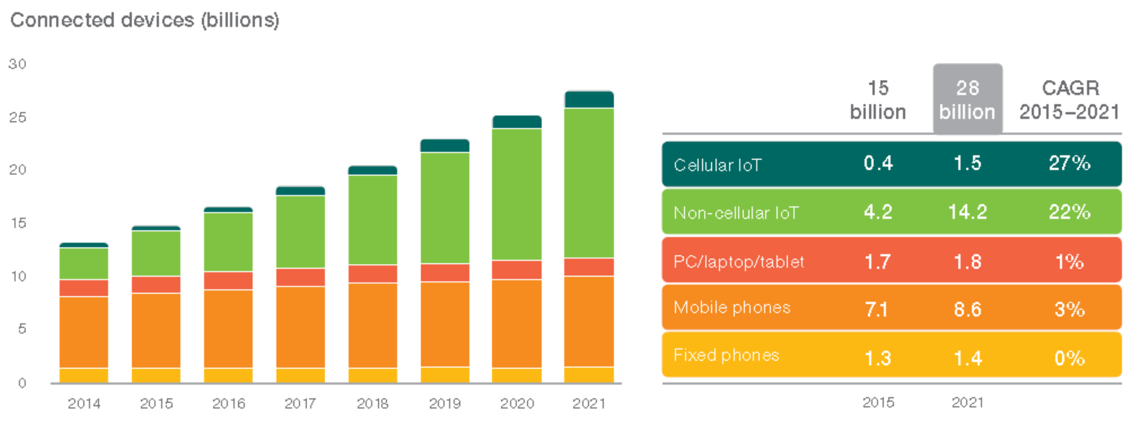


FIGURE 2. IoT economic Potential (Ip, 2016)

However, Ericsson (Richard Möller, 2016) focused its predictions on the number of sensors and devices expected to be connected to the Internet by 2021. In the report, it was stated that by 2018 the number of IoT sensors and devices will exceed the number of mobile phones and by 2021 about 28 billion devices will be connected, of which about 16 billion will be IoT related. Figure 3 shows an infographic comparing a cellular IoT, non-cellular IoT, PC/laptop/tablet, mobile phones, and fixed phones connection devices between the years 2015 and 2021



	15 billion	28 billion	CAGR 2015–2021
🟦 Cellular IoT	0.4	1.5	27%
🟩 Non-cellular IoT	4.2	14.2	22%
🟥 PC/laptop/tablet	1.7	1.8	1%
🟧 Mobile phones	7.1	8.6	3%
🟨 Fixed phones	1.3	1.4	0%
	2015	2021	

FIGURE 3. IoT connected devices are expected to surpass mobile phones in 2018 (Richard Möller, 2016)

The above-mentioned economic analysis of the IoT and the Industrial Internet of things (IIoT) and many other similar forecasts have been conducted elsewhere focusing on the economic value and the driving results of rich analytical sensor-based data sets. Moreover, aside the economic value impact of the IoT and IIoT, almost all the forecasts sorted to the mentioned areas, such as logistics, manufacturing, services and supply chain will be the core areas that can deliver the most economic value.

2 IOT ARCHITECTURE

Having realized the economic importance of IoT in the previous sections, it is considered important also to look into the technology that makes it possible to make the economic values reality. However, this chapter is dedicated to deal with the technology that will start with the IoT architecture layers. The IoT architecture layer takes a form similar to the ISO/OSI reference model ((ISO) & IEC, 1994), the Transmission Control Protocol (TCP) and Internet Protocol (IP) Suite, and the US Department of Defence 4-layer model (DoD4) (Shimonski, 2005). Table 1 illustrates the aforementioned models within the internetworking architecture.

TABLE 1. 7-layer stack and 4-layers' stacks or OSI, TCP and DoD4

OSI Model	TCP Model	DoD4 Model
Application	Application	Process
Presentation	Transport	Host-to-Host
Session	Internet	Internet
Transport	Network Access	Network Access
Network		
Data Link		
Physical		

From table 1 it can be seen that the (TCP) Internet model and DoD4 model are 4-layered and they map to each other. Moreover, the proposed IoT architecture model was based on the aforementioned model, that is, ISO, TCP and DoD4 are also a 4-layered model. In fact, based on ISO, TCP and DoD4, the IoT could have been implemented without further architectural modelling but they failed to conceive IoT features and issues such as connectivity and communications, data collection and analysis, device management, scalability, interoperability, integration and security. Thus, there was the need to restructure all the three models to conform with IoT features and issues. The IoT architecture model consists of various components and it is a 4-layer centric architecture where specific technologies can be realized at each layer. Table 2 shows the IoT 4-layered model, which shows what components are realized at each layer.

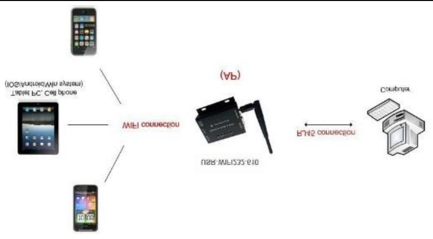
TABLE 2. IoT architectural model (Chung, 2017)

IoT Architecture layers	Components	
Application Layer	Environment, Energy, Healthcare, Transportation, People tracking, Surveillance, Supply Chain, Retail	
Management Service Layer	Device Modelling, Configuration and Management	Data flow Management, Security Control
Gateway and Network Layer	WAN (GSM, UMTS, LTE, LTE-A, 5G near future)	WiFi, Ethernet, Gateway Control
Sensors Connectivity and Network	Sensor Networks, Sensor/Actuators, Tags (RFID, Barcode)	

2.1 Sensors connectivity and network or the device layer

The layer at the bottom basically represents the IoT devices and they come in various types and forms of architecture, properties and capabilities. A device can be considered as an IoT device if such a device has any form of communication that can be connected to the Internet directly or indirectly. Table 3 illustrates some example devices that can be found at the Sensors Connectivity and Network Layer. The devices at the sensor layer have the capability to sense and collect information in real time for processing. They are of Low-Power and low data rate for connectivity. Application areas of some of these sensors can be termed as body sensor, environmental sensors and surveillance sensors.

TABLE 3. IoT sensor Layer

Sensors Layers	Technologies	Infographic example
LAN	WiFi, Ethernet	
PAN	Bluetooth, ZigBee, Z-Wave, 6LoWPAN, UWB, Wired	
Sensors or Actuators	Infrared, Solid State, GPS, Photoelectric, Accelerometer, Photochemistry, Catalytic, Gyroscope	
Tag	RFID and Barcode (1D, 2D)	

2.2 Gateway and network layer

The Gateway and Network Layer also known as the Communication Layer, supports the connectivity of the devices in a sensor or at a device layer. It consists of diverse protocols which aid in the communication between the devices and the cloud. The most notable of these protocols are the Hypertext Transfer Protocol (HTTP) with the RESTful approach, the Message Queue Telemetry Transport (MQTT) and the Constrained Application Protocol (CoAP). The IoT protocols will be studied in greater detail in subsequence chapters. Table 4 shows the Gateway and Network Layer with the technologies that are involved in it.

TABLE 4. IoT Gateway and Network Layer (Chung, 2017).

Gateway Network	WAN 3G, LTE, LTE-A, M LoRa, Sigfox, future 5G	LAN WiFi, Ethernet
Gateway	Micro-Controllers, Radio Communication Module, Signal Processor, and Modulator, Access Point, Embedded/OS, SIM module Encryption.	

Moreover, one most important aspect of the Gateway and Network Layer is its ability to aggregate data and also to host a broker communication. The broker communications and data aggregation combine communications and data from different devices and then route the information to the specific device through a gateway service (Fremantle, 2015). The Gateway and Network Layer is also capable of supporting, for example an HTTP Server and a MQTT broker to enable communications between devices. Moreover, it serves as a bridge and transforms between different protocols, such as HTTP APIs based on MQTT message to a device (Fremantle, 2015).

2.3 Management service layer

The Management Service Layer consists of two main functional parts as indicated in table 2. The two main functional parts are the Device Modelling, Configuration and Management part and the Data Flow Management and Security Control part. However, before considering the functions of the parts of the Management Service Layer, it is also important to describe what is management service. Table 5 depicts some of the services that the Management Service Layer can offer.

TABLE 5. IoT Management Service Layer components (Chung, 2017).

Management Service Layer	
Services	Components of the service
Operational Support System (OSS)	Device Management / Configuration / Management, Performance Management, Security Management
Service Analytic Platform	Statistical Analytics, Data Mining, Text Mining, In-Memory Analytics, Predictive Analytics
Billing Support System (BSS)	Billing Report
Security	Access Control, Encryption, Identify Access
Business Rules Management (BRM)	Rule Definition / Modelling / Simulation / Execution
Business Process Management (BPM)	Workflow Process Modelling / Simulation / Execution

As illustrated in table 5 the Management Service Layer has important roles in the IoT architecture. The roles can be grouped into two parts. The data service management is in charge of processes, such as information analytics, security control, process modelling and device management. The data management has two forms of techniques, the Periodic and Aperiodic data management schemes (Chung, 2017).

In the Periodic IoT data management information or data is collected periodically by an IoT sensor for an analysis. For instance, a temperature sensor monitor will record a number of information about the weather or a condition of an industrial machine within a certain period of time. However, not all gathered information gathered will be necessary for an analysis, hence a refining of the data collected by the sensor is required to filter out the unwanted and to keep the ones needed for the actual purpose of collecting the data.

In the Aperiodic data collection technique an IoT sensor collects data and requires an immediate response or attention on the information as soon as the event happens. For example, if an IoT sensor device is monitoring a patient if security is monitored, the delivery of the information should be immediate and would require an immediate response as well. Beside the data management unit, there is also the data management unit which provides management on data information flow, information access, integration and data control (Chung, 2017). There is also the data abstraction unit which provides services, such as information extraction processing, and can be used as a common business mode.

2.4 Application layer

The IoT Application Layer is the topmost layer and it is the layer that serves as an interface between the sensor application and the end users. It constitutes of various applications sectors such as environmental, industrial, healthcare, smart home asset tracking, and several others as illustrated in table 6 below. It is also a layer that hosts the IoT Application Layer protocols such as the Hypertext Transfer Protocol (HTTP), MQTT, CoAP, Advanced Message Queuing Protocol (AMQP), Extensible Messaging and Presence Protocol (XMPP), Simple Object Access Protocol (SOAP). The first three above-mentioned IoT protocols will be dealt with in detail in the following sections.

Moreover, since different applications from diverse industries and sectors are having different protocols and classifications based on the type of network, coverage area, size, business model, real time or non-real-time systems, the Application Layer protocols are able to allocate, link and exchange data or information among other application systems. The IoT classification is based on application domains, such as Personal and Home, Enterprise, Utility and Mobile. These classifications define the size of an application domain and also determine the characteristics of it.

For instance, the Personal and Home application domain represents a small scale. This mean a limited number of users, individuals or home. The enterprise IoT represents a large scale of users, in a community level. The utility IoT represents a much larger scale of users such as a national or regional of IoT support and the Mobile IoT, which are usually spread across other domains due to their mobility nature and the devices involves are mostly battery operated and portable. Table 6 illustrates some of the main application domains and market areas and sectors that the Application Layer can host.

TABLE 6. IoT Application Layer

Application Layer	
Application Sectors	A' plication Domain Smart Environmental, Smart Energy, Smart Transportation, Smart Healthcare, Smart Retail, Smart Industry, Smart Military applications
Market Areas	Supply Chain, People Tracking, Asset Management, Fleet Management, Surveillance

3 IOT GATEWAY PROTOCOL AND IP STACK

To start with, like in any other form of communication between Human-to-Human (H2H) or D2D (Device-to-Device) there should be a protocol that promotes or helps individuals or devices to understand each other. In the case of the IoT communication between D2D or (Machine-to-Machine) M2M and the cloud, there is a broad set of protocols that facilitate communications. Table 7 shows the protocol stack of the IoT in comparison with the ISO/OSI model and the (TCP) protocol stack.

TABLE 7. IoT Protocol Stack

ISO/OSI Reference Model	IoT Protocol Stack	TCP Protocol Stack
Application Layer Presentation Layer Session Layer	Application Layer Protocol HTTP/REST, CoAP, XMPP, AMQP, MQTT, DDS, SNMP, DNP, SSH, IPfix, EBHTTP, DLMS, MODBUS, NTP, LTP	Application Layer
Transport Layer	Transport Layer Protocols TCP, MPTCP, UDP, DCCP, SCTP, TLS, DTLS	Transport Layer
Network Layer	Network Layer IPv4/IPv6, 6LoWPAN, ND, DHCP, ICMP	Internet Layer
Data Link Layer Physical Layer	Physical Layer 3GPP MTC. IEEE 802.11 Series, IEEE 802.15 Series, 802.3, 802.16, WirelessHART, Z-WAVE, UWB, IrDA, PLC, LonWorks, KNX	Link Layer

The focus of this thesis is to discuss in detail the applications of the main and most well-known IoT potential protocols at the Application Layer. The three most popular IoT protocols which this thesis work is studying are summarized in the table 8 below.

TABLE 8. *A summary of IoT Application Layer protocols*

Protocol	Transport Protocol	Messaging	WAN (2G, 3G, 4G)	Power	Compute Resources	Security
HTTP/ REST	TCP	Rqst/Rspnse	Excellent	Fair	100Ks/RAM Flash	Low-Optimal
MQTT	TCP	Pub/Subsrb Rqst/Rspnse	Excellent	Good	10Ks/RAM Flash	Medium-Optimal
CoAP	UDP	Rqst/Rspnse	Excellent	Excellent	10Ks/RAM Flash	Medium-Optimal

The following sections of the chapter will be a presentation of the details of the above summarized IoT protocols.

3.1 Hypertext transfer protocol (HTTP)

HTTP (Fielding & Reschke, 2014) is the most widely and popularly adapted Application Layer protocol on the World Wide Web. The standardization of HTTP has been done by the Internet Engineering Task Force (IETF) in collaboration with the World Wide Web Consortium (W3C) (MIT). HTTP works on a Client-Server messaging technology where the client requests for a Hypertext Markup Language (HTML) page from a server and the server also responses with an HTML page. As illustrated in table 8 HTTP relies on the TCP as a transport protocol, which uses sockets to transfer data. The connection between the client and server begins with the client via a socket connection on the port 80, which is the assigned port number for HTTP to the Server. When the connection is established, it means that the server accepts the request of the client, which is in an HTML page form, and other objects.

However, upon the connection establishment, the HTML pages and the objects are then exchanged between the client browser and the web server. After the completion of the request, the TCP terminates the connection between the client and the server and also clears the memory so that previous requests from the client are removed. With the HTTP, requests, such as GET, PUT, POST and DELETE, are the four methods mostly used. The GET request displays a web page and its objects upon a request to the user. The PUT and POST request methods are used to modify server resources and the DELETE request removes resources that are not needed.

Moreover, there are two HTTP connection types that can be established with the TCP. These are the Non-Persistent (HTTP/1.0) and Persistent (HTTP/1.1) connections. The main difference between the Non-Persistent (HTTP/1.0) and Persistent (HTTP/1.1) depends on the number of TCP connections needed to transmit a Uniform Resource Locator (URL) of a web page and its objects. Figure 4 shows an HTTP connection scheme between a client and a server.

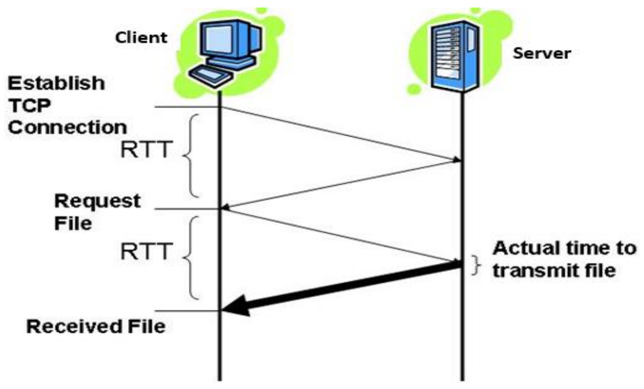


FIGURE 4. HTTP establishing TCP connection between Client and Server (WIKIBooks, 2015)

In figure 4 the Round-Trip Time (RTT) is the time spent to send a packet from a client to a server and to get a response. It also represents the time required to establish a TCP connection, send a request, and get a response or receive a file with its transmission time. Mathematically, the total RTT, from the beginning of TCP connection establishment to the receiving of the file requested, can be expressed as 2 x RTT + File Transmission Time (FTT).

3.2 Representational state transfer (REST)

REST is a language and operating system independent software architecture for designing network applications and distribution of an HTTP system to connect machines together. REST is a stateless, Client-to-Server, cacheable, point -to-point with uniform interface and its designed as a lightweight system (Vermesan Ovidiu, June 1, 2014). The communication mode begins when a client sends a message in the form of a request to a server and the server replies back to the client in a response form indicating whether the request sent by the client was successful or whether there was an error. With REST, the communication between devices to the cloud is possible over the TCP/IP where an HTTP is used to connect to the world wide web (www).

4 COAP PROTOCOL

CoAP (Shelby, Hartke, & Bormann, Internet Engineering Task Force (IETF), 2014) is an Internet based Client-to-Server model document transfer protocol similar to HTTP and it has been standardized within the IETF, the Constrained RESTful Environments (CoRE) working group (Bormann, Jimenez, & Melnikov, 2010). It is designed for constrained devices and constrained networks. Constrain devices are embedded devices with limited power, memory and processing resources and they are expected to be connected and function similar to mainstream processes. HTTP is the main protocol because the connectivity between a client and server is too heavy for such devices. CoAP was developed to address the limitations HTTP has over constrained devices, such as sensors and devices with Low-Power connected via Lossy Networks (LLNs).

The design model of CoAP is equivalent to HTTP Client-to-Server model but most of its implementation is within M2M or D2D communications and they can act both as a client and a server role. CoAP does not support the Transmission at Transport Control Protocol but it runs over the User Datagram Protocol (UDP). It utilizes the UDP broadcast and multicast for addressing and the interaction between a client and server is asynchronous over the UDP. However, since the datagram-oriented transport is connectionless, the Client-to-Server communication of CoAP is also connectionless and it can be used on top of Short Message Services (SMS) and other Packet based communication protocol.

Moreover, devices connected with CoAP have the ability to discover and explore each other to negotiate ways to exchange data among themselves. CoAP also supports the observe resource state changes methodology. It is a state transfer model which allows a client to continuously receive responses from a server. This is important for example in an IoT healthcare application where data from a sensor attached to a patient is vital and needs to be monitored constantly. In other words, CoAP is an asynchronous message exchanger which happens via observe/notifications. Similar to HTTP, a client uses the GET request command in an observable mode to express interest in any updates from the server. The client receives a notification each time the state of the resource changes at the server.

CoAP is also designed as a Conditional Observer (Technical Report - Protocol Analysis , 2014) or an event-based model. This means that it allows the client to be notified only when certain actions on the observed resources are met. Since messages are not received at every event that occurs but only at events that are needed, energy can be save due to the control messages received. For instance, in an IoT application for temperature monitoring, a sensor may send an update every second, even though nothing significant has changed from one data transfer to the other. With CoAP observed resources, only interesting events that happen periodically or an observed value changes with a pre-specified step size will be notified. Another feature of CoAP is that it supports proxies, i.e., a client can request data from a CoAP server with HTTP requests.

4.1 CoAP message types

In the course of exchanging messages within the CoAP Network, there are four defined message types. These are Confirmable, Non-Confirmable, Acknowledgement and Reset.

When a client sends a request to a server with Confirmable Messages (CON), it requires that the receiving end will acknowledgement (ACK) the message with the same message ID. This transmission between a client and server is usually used when a reliable delivery of a data is required. A retransmission of the data occurs after a waiting time for an ACK elapses and it will repeat the circle until an ACK is received with the message ID. Figure 5 shows a reliable message transmission between a client and server.

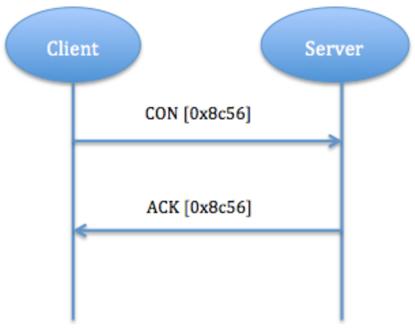


FIGURE 5. CoAP Confirmable message transmission (Chen, 2014)

A Non-Confirmable data transmission technique does not require ACK and it is unreliable. This type of data exchange technique uses a NON-message type, which contains a message ID to supervise the transmission. It is most prevalent with data streams where data is sent and there is a possibility that data is lost or received out of order during the transmission. Figure 6 shows Non-Confirmable message transmission between a client and server.

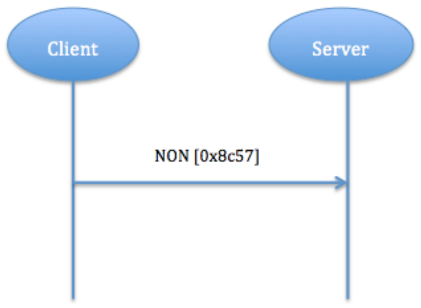


FIGURE 6. CoAP Non-confirmation message transmission (Chen, 2014).

Moreover, as depicted in figure 5, an ACK message with a message ID is sent to the client (sender) from the server (receiver) that a specific Confirmable message (CON) has arrive.

CoAP supports piggybacked messages too. When a client sends a request using CON type or NON-type messaging it receives an ACK message immediately if it is a Confirmable message. The ACK contains a response message of successful or failed delivery of the sent message. Again, when a server receives a CON message request and it is unable to response the request immediately, it sends an empty ACK so that in case a client will resend the message after certain time elapses. However, a new CON is sent to the client whenever the server is ready to response to the message and the client replies with an ACK to confirm the CON message from the server. Figure 7 shows separate responses when a client used the GET request to request a temperature from a client.

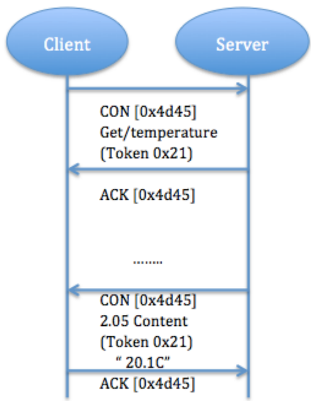


FIGURE 7. CoAP CON request message with separate responses (Chen, 2014).

Therefore, an ACK message of Confirmable messages does not indicate success or failure of any request, but an ACK message may also carry a piggybacked response.

The Reset also known as the Negative Acknowledgement (NACK) is an error message sent from a receiving end (Server) to notify the sender (Client) that a specific message is lost or the receiver failed to process the message. In other words, a reset message is sent to reject error and unknown messages when specific messages (Confirmable or Non-Confirmable) are received but some context is missing to be properly processed.

4.2 CoAP message format

CoAP is based on the exchange of compact messages that by default are transmitted over UDP. Messages of CoAP are encoded in a simple binary format and it is a fixed-size 4-byte header followed by optional extensions such as a variable -length Token Value, a sequence of zero or more CoAP options in the Type-Length-Value (TLV) format and an optional payload that takes up the rest of the datagram. Table 9 shows the structure of CoAP message format. The 4-byte header consists of the Version (Ver, 2-bit), Type (T, 2-bits), Token Length (TKL, 4-bits), Code (8-bit) and a message ID (8-bits).

TABLE 9. CoAP message format

Ver (2bits)	Type (2bits)	TKL (4bits)	Code (8bits)	Message ID (16bits)
Token 0-8 bytes (if any, indicated by TKL)				
Options (if any)				
Payload (if any)				

The 2-bit unsigned integer version file specifies the CoAP version number and for RFC-7252 CoAP specification implementation it is set to 1 (01 binary). This means that every message must have this version number. Otherwise messages with an unknown version number are silently ignored. Other values are reserved for future versions.

Type (T) is also a 2-bit unsigned integer within the header that indicates whether a message is of type Confirmable, Non-Confirmable, Acknowledgement, or Reset. The Token Length is a 4-bit unsigned integer within the header that indicates the length of the variable-length Token field, which is between 0 and 8 bytes. If the number is set to 0, it means that there are no options and the payload (if any) follows immediately the header. However, if the number is greater than 0, the field indicates the number of options to immediately follow the header.

Code is also an 8-bit unsigned integer within the header and it is split into subfields; a 3-bit class (most significant bits) and a 5-bit detail (least significant bit). The class can indicate a request, a successful response, a client error response or a server error response. The message ID is a 16-bit unsigned integer within the header, too, and it is used to detect a message duplication and to match messages of the type Acknowledgement/REST to messages of the type Confirmable/Non-Confirmable.

The Token Value is next to the header and it is 0 to 8 bytes as given by the Token Length field within the header. It is used to correlate requests and responses. However, the 8-byte long header help to protect attacks, such as spoofing, and it is a rule that all CoAP messages have Tokens even if they have zero-length.

As stated earlier, CoAP options may only be present if the variable-length Token field value is a non-zero. The option field holds information that affects the performance and functionality of the CoAP. Moreover, CoAP defines the number of options that can be included in a message and each option instance in a message specifies the option number, the length of the option and the option value itself. Details and an exhausted list of options are elaborated in the RFC-7252 documentation (Shelby, Hartke, & Bormann, 2014). The payload is also optional and can only be available when it is non-zero-length and prefixed by a one-byte payload marker (0xFF), which indicates the end of options and the start of the payload.

Payload Data extends from after the marker to the end of the UDP datagram. An absence of the payload marker represents a zero-length payload and the presence of a marker on other hand, followed by a zero-length payload, must be processed as a message format error. Moreover, the request and response messages from client and server respectively can contain payload data. It can also be carried along with a Confirmable message and a Non-Confirmable message. It can also be Piggybacked on Acknowledgement messages.

4.3 Message transmission between client and server

Exchanges of messages between CoAP endpoints (Client-to-Server) are performed asynchronously. CoAP uses the UDP protocol for transporting messages. It is bound to be unreliable, which means that messages may arrive out of order, may appear duplicated or may go missing without noticing.

However, CoAP implements a lightweight reliable mechanism similar to the TCP protocol, that has features such as;

- Simple stop-and -wait retransmission reliability with exponential back-off for Confirmable messages
- Duplicate detection for both Confirmable and Non-Confirmable messages.

Messages transmitted within CoAP use Request and Responses transmission techniques. Details of these transmission techniques are explained in the following sections.

4.3.1 Requests

The request methods of CoAP are similar to that of HTTP request methods of GET, POST, PUT and DELETE. The GET method is used to retrieve the state of information resource, which is given in the Uniform Resource Identifier (URI). Information, such as sensor values, for example temperature, device names or a state of a device, can be retrieved by the GET method. The POST and PUT methods are similar in operation, they are simply used to create a new resource or when a target resource is updated. The DELETE method request is used to remove the resource specified by the URI.

As stated earlier in this chapter CoAP supports observe method of resource changes and it is another form of request method for the client to use to observe a resource over a certain period of time. This method was designed because the GET, POST, PUT and DELETE methods do not work well when a Client wants to observe a resource from a server. The observe method allows the CoAP sever node to send notifications continuously after it has received a registration message from a client. The aim of the server is to keep the client updated by notifying the observer (client) of the latest resource values.

When an observer (client) is interested in observing a resource, it sends a registration message to the server. The message sent applies the GET request with an observe option value set to “0”. The server adds the client to the list of observers of the resource and starts sending notifications. The notification messages have a set value in the observe field and they are used to check the updated measurement. In the case where a server is not able to add a new observer, it sends a response without the observe option value. Figure 8 taken from RFC7641 (K., 2015) shows how a client registers its interest in a resource and receives a notification. In this example, a client is interested in observing the temperature at the server and starts by sending a registration message to the server. The server adds the client (observer) to its database and starts sending notifications to the observer. When the client no longer interested in observing the temperature, it sends a deregistration message with an observer option set value “1”.

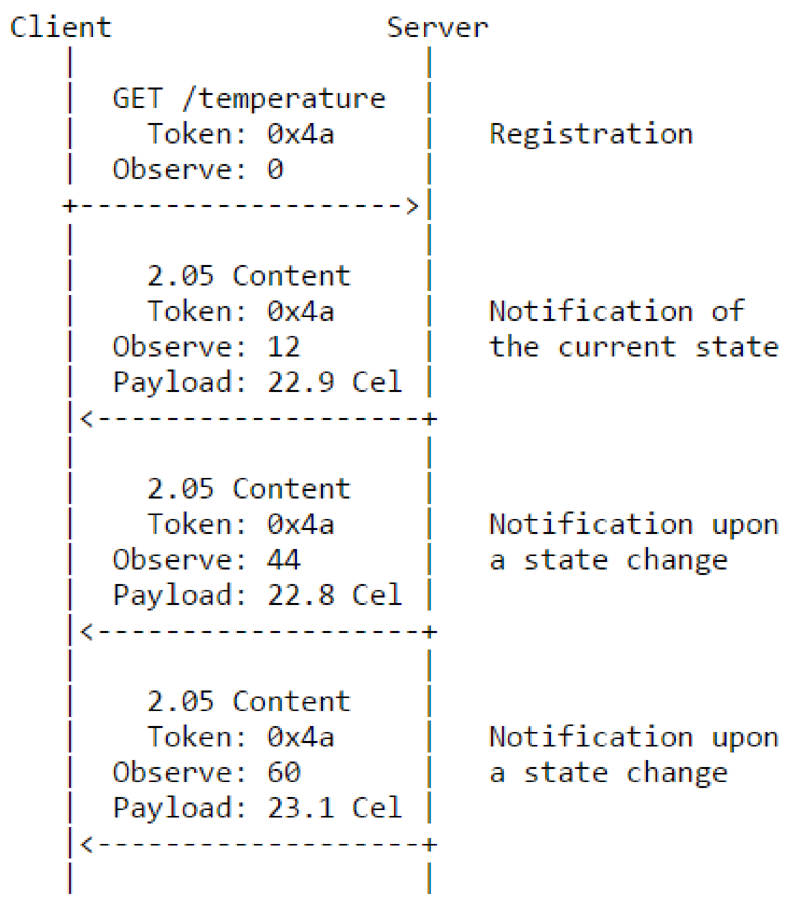


FIGURE 8. Client observing a resource in CoAP from RFC7641 (K., 2015)

Another way for a client to stop observing is to reject a notification by sending a Reset message. Also, the transmission messages between the server and the client could be a Confirmable or Non-Confirmable. In the case of Confirmable message, the server will expect an Acknowledgement from the client. If after a defined period of time with several retransmissions, it did not receive the Acknowledgement, the server consider that the client is no longer interested in observing the resource and then removes the client from the observer list.

4.3.2 Response

When a request is sent from a client to a server, the server responds with a matching request by means of the client Generated Token. A response is identified by the Code field in the CoAP header being set to a Response Code. The following are the Response Code classes within the CoAP:

4.3.2.1 Success 2.xx

This class or Response Code indicates that the client request was successfully received, understood and accepted.

4.3.2.2 Client error 4.xx

This class of Response Code is intended for cases in which the client seems to have an error. This Response Code applies to any request method.

4.3.2.3 Server error 5.xx

The server error class Response Code indicates cases in which the server is aware that it has an error or is incapable of performing the request. These Response Codes are applicable to any request method.

4.4 CoAP security

As in any communication between devices, security is important and it is not different in the CoAP protocol which has been a standard (ISO/IEC 20922) (Richard J Coppen, 2016) for IoT applications. However, when considering security of any communication systems, there are three elements that should be considered. These are the system integrity, authentication and confidentiality. The Datagram Transport Layer Security (DTLS) RFC 6347 (Rescorla E., 2012) has been developed as a security protocol for CoAP. First of all, CoAP uses a datagram transport and DTLS can achieve the above-mentioned security elements. It is well suited for securing applications and devices that are delay sensitive, has the mechanism of reordering messages which are arriving out of order, retransmission of lost messages during the handshake and message sizes. It is tolerant to errors during decryption but no error messages and no session termination. It also adds three implements: 1 Packet retransmission, two assigning sequence number within the handshake and three replay detections.

The DTLS is composed of two layers. The lower layer, known as the DTLS Record Protocol, provides connection security and it has two basic properties:

- ➢ Connection is private by using a symmetric encryption

- Connection is reliable by including a message integrity check.

These properties or options may be used alone, together or not at all.

The upper layer is composed of three protocols which include Alert, Handshake and application Data.

- The DTLS Handshake Protocol is used to negotiate the security parameters of a session later used for protected communication.
- The DTLS Alert Protocol can be used at any time during the handshake and up to the closure of a session, signalling either fatal errors or warnings.
- The DTLS application Data Protocol is composed by the application Data messages that are carried out the record layer and are fragmented, compressed and encrypted based on the current connection state.

Moreover, in some conditions, the Change Cipher Spec Protocol may replace one of the above mentioned DTLS security protocols. The Change Cipher Spec message protocol is used to notify the Record Protocol to protect subsequent records by using negotiate cipher suite and keys. Figure 9 illustrates the process of DTLS Handshake protocol.

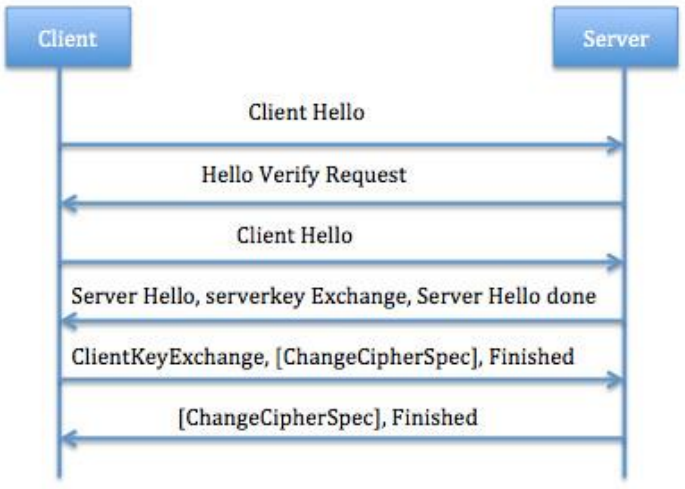

Client
Server
Client Hello
Hello Verify Request
Client Hello
Server Hello, serverkey Exchange, Server Hello done
ClientKeyExchange, [ChangeCipherSpec], Finished
[ChangeCipherSpec], Finished

FIGURE 9. DTLS Handshake process (Chen, 2014)

5 MQTT PROTOCOL

The MQTT (ISO/IEC 20922) (Richard J Coppen, 2016) is a machine-to-machine (M2M) IoT connectivity transport protocol suitable for Low-Power and Lossy Networks. MQTT is designed as a Client-to-Server and it employs a publish/subscribe messaging protocol paradigm. Its implementation is based on the TCP/IP protocol, which is characterized as reliable, ordered and error-checked protocol. It is extremely light weight, open, simple and easy to implement. It is designed to provide connectivity to embedded devices, to enable communication within constrained environments, i.e. communication in M2M and IoT devices and applications where a small Code footprint is required or the Network bandwidth is limited.

MQTT uses the publish/subscribe architecture which consists of the Publisher (Client), Subscribers, a Broker (Server), Sessions and Topics. The publish/subscribe paradigm is a communication protocol between a client and server/subscriber which requires a central MQTT Broker to manage and route data among MQTT networks nodes or subscribers. The publishers are lightweight sensors that connect to a Broker to send data. Subscribers are devices or applications that are logically attached to a client who is interested in a sensor data and they are connected to the Broker to be informed whenever new data is received. The Broker classifies sensor data into topics and sends them to the subscribers interested in the topics. Technically, topics are message queues that support the publish/subscribe pattern for clients and logically, topics allow clients to exchange information with defined semantics.

Finally, a session identifies an attachment of a client to a server. All communication between client and server takes place as part of a session. Figure 10 illustrates the MQTT data transmission architecture with a Broker, which serves a data server directing all data to their appropriate destinations.

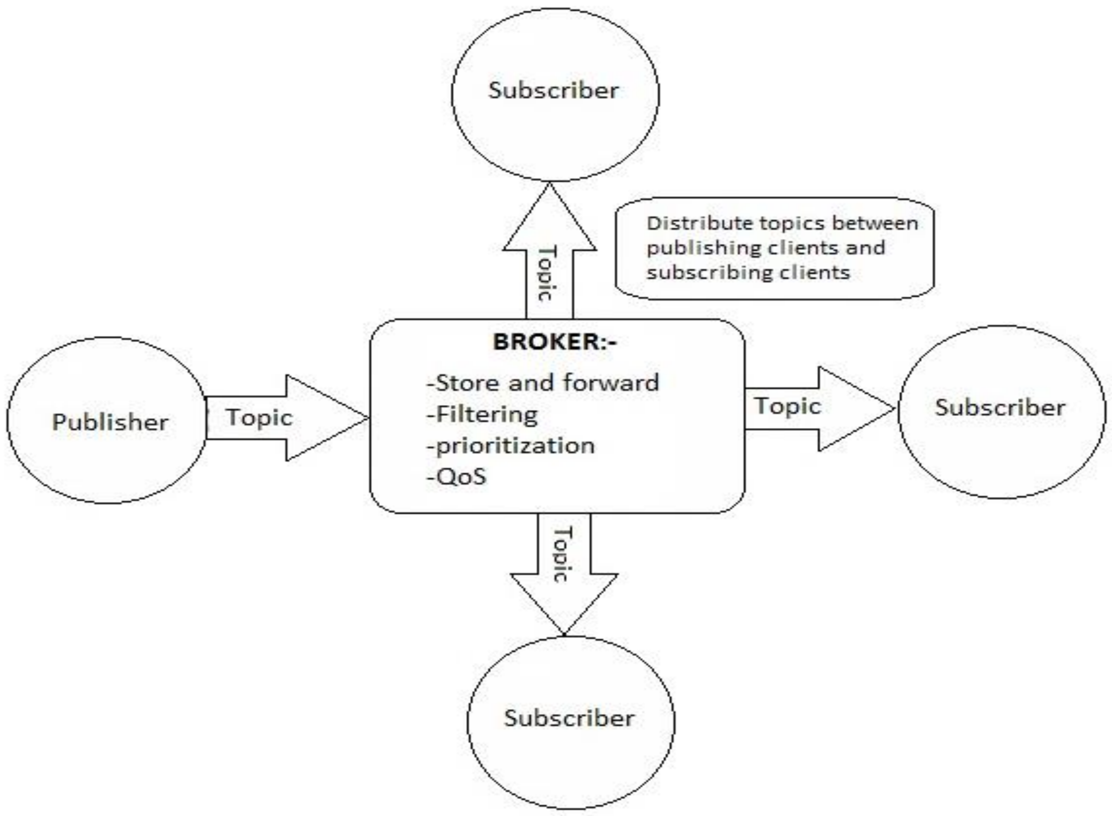


FIGURE 9. MQTT data transmission architecture with a broker

As depicted in figure 10, it can be deduced that the publish/subscribe message pattern provides a one-to-many messaging and that the Broker controls the distribution of information between the publisher client (the source of data) and the subscriber client (the destination of the data). The Broker stores, forwards, filters and prioritizes published requests from the publisher client to the subscriber clients. With the MQTT Broker system, clients can switch between the publisher and subscriber roles depending on their objectives at a particular instance. Also, within the Broker there are the MQTT Quality of Service (QoS) levels. The QoS levels are of 0, 1 and 2 that describe the increasing levels of the guaranteed message delivery.

5.1 MQTT messaging

MQTT defines fourteen (14) different messaging methods. The main messaging types that end users only need to employ are the connection request to the server

(CONNECT), DISCONNECT, SUBSCRIBE, UNSUBSCRIBE and PUBLISH messages. The other message types are used for internal mechanisms and message flows. Table 10 shows a list of the messaging types.

TABLE 10. MQTT Messaging types

Enumeration	Mnemonic	Description
0	Reserved	Reserved
1	CONNECT	Connection request to Server
2	CONNACK	CONNECT Acknowledgement
3	PUBLISH	PUBLISH message
4	PUBACK	PUBLISH Acknowledgement
5	PUBREC	PUBLISH Received (assured delivery part 1)
6	PUBREL	PUBLISH Release (assured delivery part 2)
7	PUBCOMP	PUBLISH Complete (assured delivery part 3)
8	SUBSCRIBE	SUBSCRIBE request
9	SUBACK	SUBSCRIBE Acknowledgement
10	UNSUBSCRIBE	UNSUBSCRIBE request
11	UNSUBACK	UNSUBSCRIBE Acknowledgement
12	PINGREQ	Ping Request
13	PINGRESP	Ping Response
14	DSCONNECT	Client Disconnecting
15	Reserved	Reserved

5.1.1 Connect and subscribe messaging explained

Figure 11 illustrates the connection session and subscription setup between a client and a server with a clean session flag set 1 (flag =1).

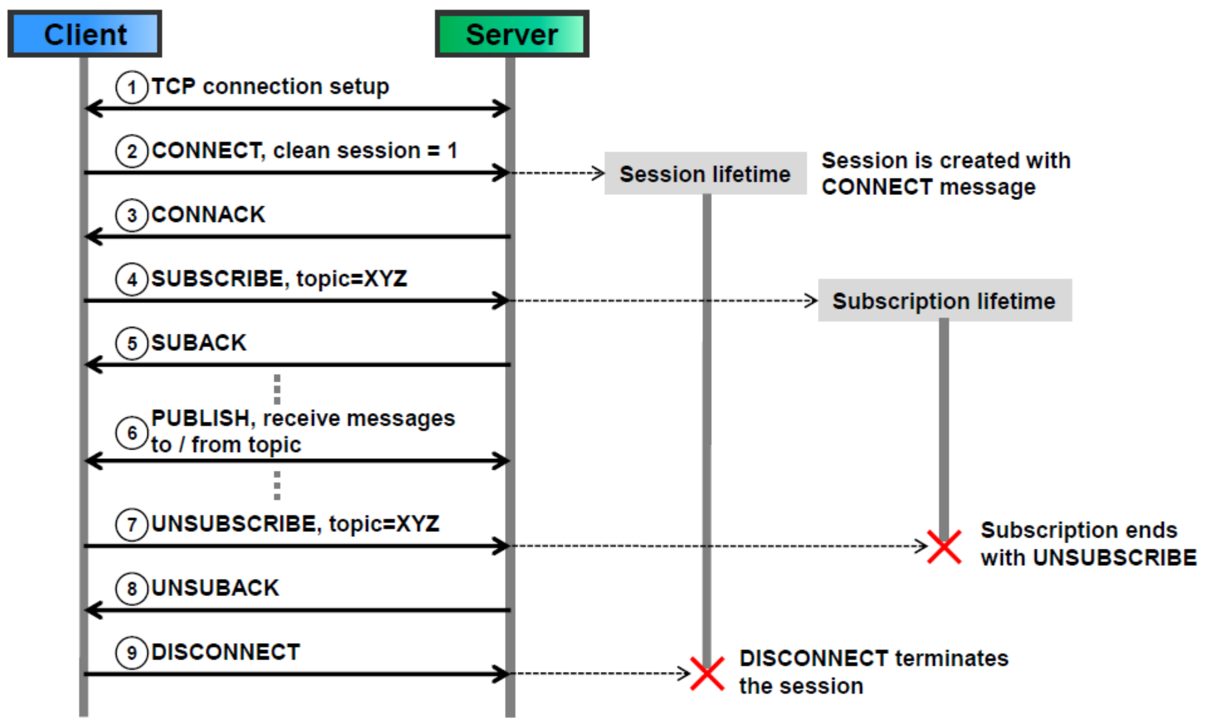

Client
Server
1 TCP connection setup
2 CONNECT, clean session = 1
Session lifetime
Session is created with CONNECT message
3 CONNACK
4 SUBSCRIBE, topic=XYZ
Subscription lifetime
5 SUBACK
6 PUBLISH, receive messages to / from topic
7 UNSUBSCRIBE, topic=XYZ
Subscription ends with UNSUBSCRIBE
8 UNSUBACK
9 DISCONNECT
DISCONNECT terminates the session

FIGURE 10. MQTT CONNECT and SUBSCRIBE messaging

Figure 12 illustrates the session and subscription setup between a client and a server with a clean session flag set 0 (flag = 0).

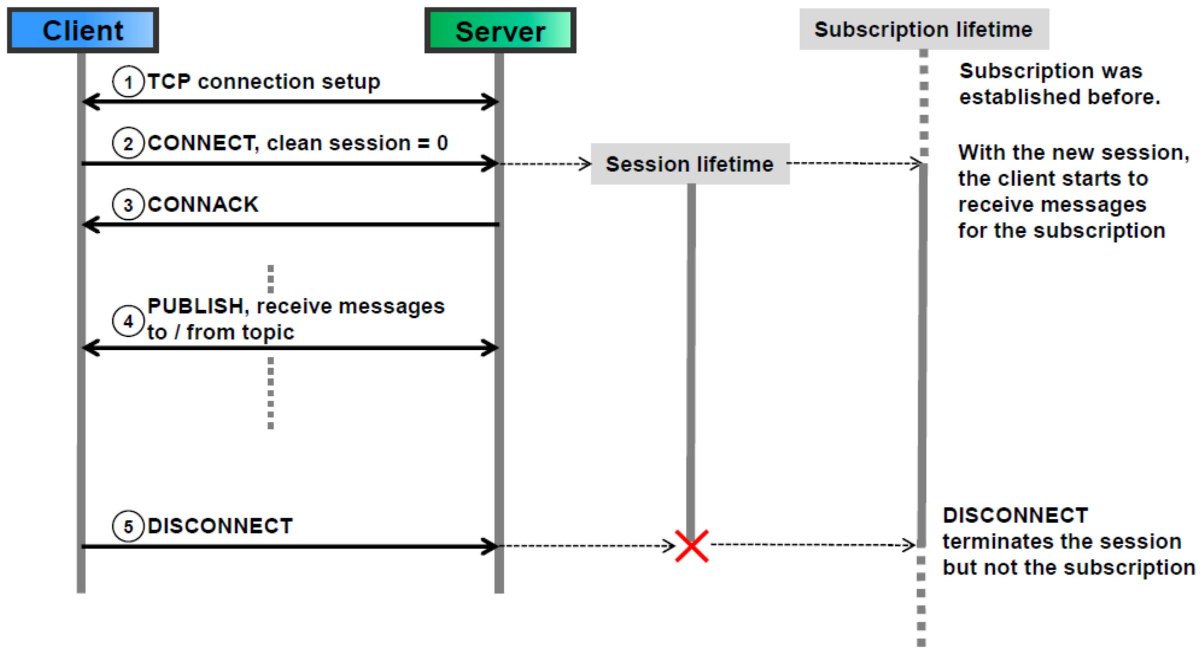

Client
Server
Subscription lifetime
Subscription was established before.
1 TCP connection setup
2 CONNECT, clean session = 0
Session lifetime
With the new session, the client starts to receive messages for the subscription
3 CONNACK
4 PUBLISH, receive messages to / from topic
5 DISCONNECT
DISCONNECT terminates the session but not the subscription

FIGURE 11. MQTT CONNECT Subscription messaging (flag = 0)

5.2 MQTT messaging formats

The MQTT protocol messaging format also known as the control Packet, consists of three parts; Fixed Header, Variable Header and Payload. Every MQTT control Packet contains a Fixed Header. It consists of two bytes. The first byte contains the Message Type and the Flags that have fields such as the Duplicate flag (DUP), QoS level and RETAIN. The second field contains the Remaining Length field. Table 11 illustrates the Fixed Header fields.

TABLE 11. MQTT Fixed Header format

Field Length (bits)	7	6	5	4	3	2	1	0
Byte 1	Message type				DUP flag	QoS level		RETAIN
Byte 2	Remaining Length (1-4 bytes)							

As depicted in table 11 byte 1 consists of the Message Type and what is term as Flags (DUP, QoS level and RETAIN) fields. The second byte (byte 2), the Remaining Length field has at least one-byte. Further description of the MQTT messages in the Fixed Header fields are explained in table 12.

TABLE 12. MQTT message Fixed Header field explained

MQTT Message Fixed Header field	Description values	
Message Type	0: Reserved	8: SUNSCRIBE
	1: CONNECT	9: SUBACK
	2: CONNACK	10: UNSUBSCRIBE
	3: PUBLISH	11: UNSUBACK
	4: PUBACK	12: PINGREQ
	5: PUBREC	13: PINGRESP
	6: PUBREL	14: DISCONNECT
	7: PUBCOMP	15: Reserved
DUP (Duplicate) Flag	A Client or a Server (Broker) attempt to re-delivers a PUBLISH, SUBSCRIBE or UNSUBSCRIBE message. The Duplicate (DUP) bit is set as a message flag to indicates to the receiver a message may have already been received. This applies to messages with a QoS value greater that zero (0).	
QoS level	This indicates the level of delivery assurance of a PUBLISH message. Level 0: At most once delivery, no guarantee. Also, known as Fire and Forget Level 1: At least once delivery and with acknowledged delivery Level 2: Exactly once delivery with assurance of delivery Level 3: Reserved	
RETAIN	It instructs the Server (Broker) to RETAIN the last received PUBLISH message and deliver it as a first message to a new subscription after it has been delivered to the current subscribers. This is possible when the RETAIN flag is set to one (1).	
Remaining Length	It indicates the number of remaining bytes in the current message, including Data in the variable header and the payload.	

5.3 MQTT QoS

MQTT provides the typical delivery of QoS levels of message oriented middleware. Even though the TCP/IP on which MQTT reside provides a guaranteed data delivery, however, data loss can still occur during the data transmission if the TCP connection breaks down. Therefore, MQTT adds three (3) QoS levels on top of the TCP.

5.3.1 QoS level 0

At Most Once delivery (Fire and Forget). With this QoS level, messages are delivered in accordance to the delivery guarantees of the underlying TCP/IP Network. No PUBACK is expected and no retry semantics are defined in the protocol. The message is delivered to the Server or not at all. An example of application scenario could be a temperature sensor. Temperature sensor data is published regularly and loss of any individual value is not critical since subscribers to the temperature data integrate lots of sample values over time and hence an individual sample does not make any difference. Figure 13 illustrates a published message flow with QoS level 0 delivery semantics.

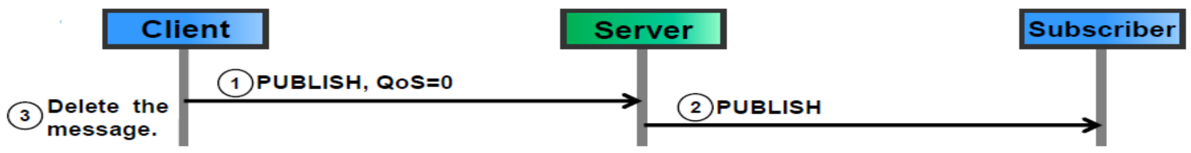


FIGURE 12. QoS Level 0 At Most Once delivery semantics

5.3.2 QoS level 1

At least once delivery. With this delivery semantics, messages are guaranteed to arrive at the server and should be acknowledged (PUBACK). However, there could be a Duplicate (DUP) which could arise due to a delay in the arrival of an Acknowledgement (PUBACK) or an identified (ID) failure of either the communications link or the sending device. This means that when a sender (client) PUBLISH is data, after sometime if PUBACK is not received, it resends the data again with a DUP bit set in the message header resulting in duplication of messages. However, the application can discard a Duplicate message by evaluating the message ID field. An application scenario could be a sensor monitoring the state of a door. This means that a door state is either OPEN/CLOSE or CLOSE/OPEN and these changes of states are published To Subscribers, For Example In The Form Of An Alarm Or Beacon A Buzzer. Figure 14 below shows the QoS level 1 delivery semantics

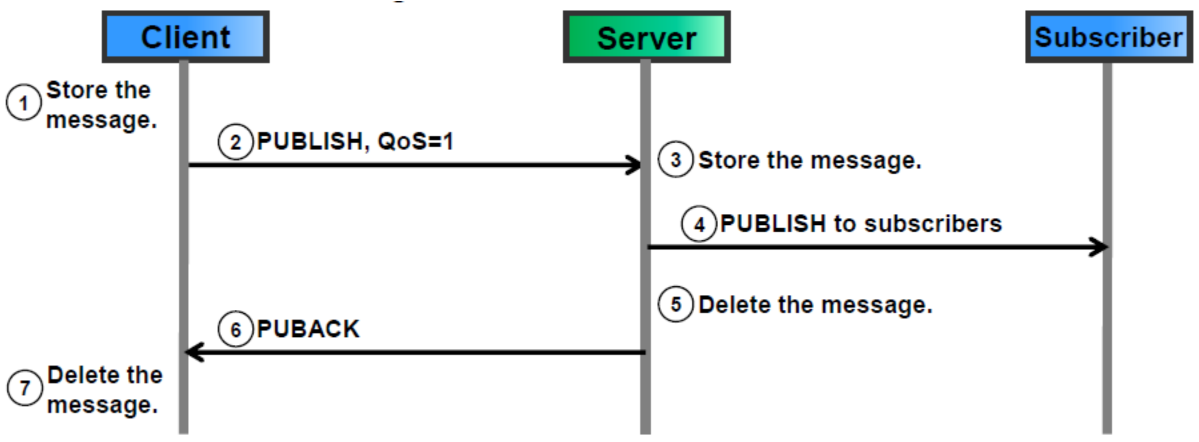


FIGURE 13. QoS Level 1 At Least Once delivery semantics

5.3.3 QoS level 2

Exactly once delivery semantics. This is the highest, safest, and slowest and QoS level and it guarantees that each message is received only once by the subscriber. It also incurs most overheads in terms of control messages and the need for locally storing the messages. It is also a combination of **At Least Once** and **At Most Once** delivery guarantee semantics.

With QoS level 2, when a receiver received the PUBLISH message, it processes the message and acknowledges it with the PUBREC message. The receiver also stores the message with a reference to the message identifier until it has sent the PUBCOMP. This is to avoid a duplication of processing the message twice. Also, the store PUBLISH message stored at the client can be discarded after it has received the PUBREC. The PUBREC message is stored upon the arrival and the client responds with a PUBREL. The receiver on the other side also deletes all stored messages upon receiving the PUBREL and a similar event happens at the client side after receiving the PUBCOMP from the subscriber. Figure 15 explains the QoS level 2 semantics.

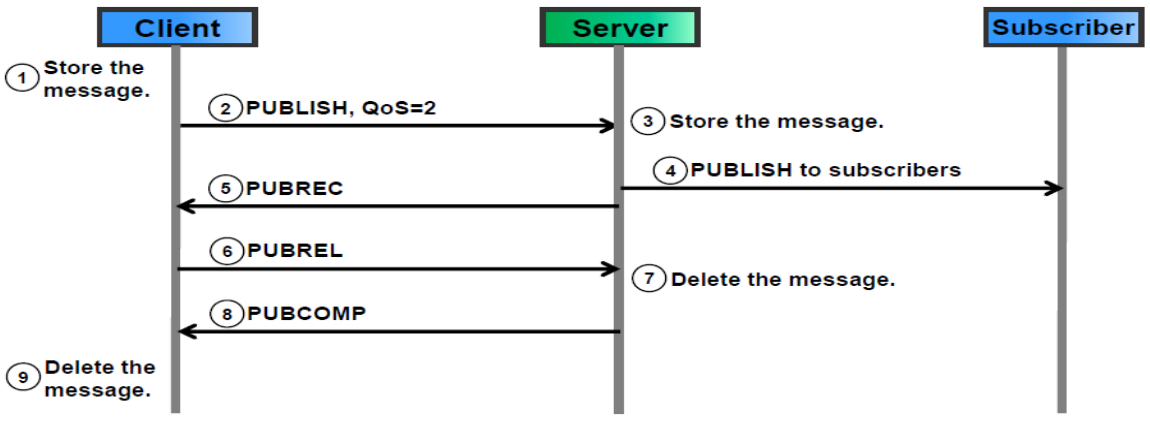


FIGURE 14. QoS Level 2 Exactly Once delivery semantics.

5.4 MQTT variable header

Some types of MQTT messages contain a variable header component. This variable header component resides between the Fixed Header and the payload (Richard J Coppen, 2016). The content of the variable header varies depending on the Packet type. The Packet Identifier field of a variable header is common in several Packet types. The component of many of the control Packet types consists of a 2-bytes Packet Identifier. These control packets are PUBLISH (where QoS >0), PUBACK, PUBREC, PUBCOMP, SUBSCRIBE, SUBACK, UNSUBSCRIBE and UNSUBACK. Table 13 illustrates the variable header format residing between the Fixed Header and the payload with the various fields included in it.

TABLE 13. Variable Header residing between Fixed Header and Payload

Field Length (bits)	7	6	5	4	3	2	1	0
Byte 1	Message Type				-	-		-
Byte 2	Remaining Length							
Byte 3	Protocol name UTF-8 encoded prefixed with 2 bytes string length (MSB)							
Byte 4	Protocol version (0x03 for MQTT version 3)							
Byte 4	Username Flag	Password Flag	Will RETAIN	Will QoS	Will Flag	Clean Session		Reserved
Byte 5	Keep Alive Timer MSB							
Byte 6	Keep Alive Timer LSB							
Byte 7	Client Identifier							
Byte 8	Will Topic							
Byte 9	Will Message							
Byte 10	Username							
Byte 11	Password							

The variable header fields are described in table 14 in which they appear in the header.

TABLE 14. MQTT Variable Header fields descriptions

CONNECT Message Field	Description / Values
Protocol Name	UTF-8 encoded protocol name string
Protocol Version	Value 3 for MQTT Version 3
Username Flag	If set to 1 it implies that payload contains a username
Password Flag	If set to 1 it implies that payload contains a password. That is if username flag is set, password flag and password must as well be set
Will RETAIN	If set to 1 it indicates or inform the Server that a Will Message should be retain for the Client which is published in case the Client disconnects unexpectedly
Will QoS	It specifies the QoS level for the Will Message
Will Flag	It indicates that the message contains a Will Message in the Payload along with retain and will QoS Flags
Clean Session	If set to 1, the Server discards any previous information about the re-connecting Client (clean new session). If set to 0, the Server keeps the subscriptions of a disconnecting Client including storing QoS level 1 and 2 messages for this Client. When the Client reconnects, the Server publishes the stored messages to the Client
Keep Alive Timer	Used by the Server to detect broken connections to the Client
Client Identifier	The Client identifier (between 1 and 23 characters) uniquely identifies the Client to the Server. The Client identifier must be unique across all Client connecting to a Server
Will Topic	Will topic to which a Will Message is published if the Will flag is set
Will Message	Will Message to be published if will flag is set
Username and Password	Username and Password if the corresponding Flags are set

5.4.1 Keep alive timer

Table 13 shows other fields, such as Keep Alive Timer within the variable header of the MQTT CONNECT message. It is the maximum time interval between messages received from the client in seconds. In case there is a drop-in connection between the client and the server, it enables the server to detect the drop without waiting for the long TCP/IP timeout. Within the Keep Alive Time period, the client has to send a packet or data to the server. With the absence of data during the Keep Alive Time, the client sends a PINGREQ message to the server, which the server responds with a PINGRESP Acknowledgement.

However, after one and half (1.5) Keep Alive Time period, the server disconnects the client as if a DISCONNECT message had been sent by the client if no message has been received within the period. In addition, the client will disconnect or will end the TCP/IP socket connection if it does not receive a PINGRESP message after sending the PINGREQ message. Figure 16 shows the communication between the client and server utilizing the Keep Alive Timer with PINGREQ.

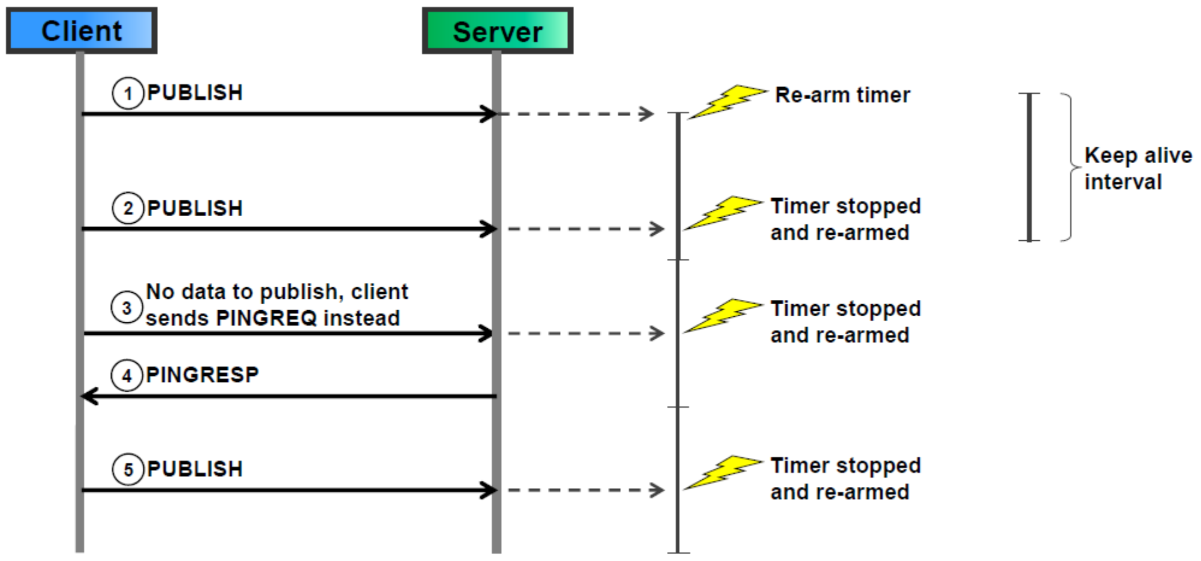

Client
Server
1 PUBLISH
Re-arm timer
Keep alive interval
2 PUBLISH
Timer stopped and re-armed
3 No data to publish, client sends PINGREQ instead
Timer stopped and re-armed
4 PINGRESP
5 PUBLISH
Timer stopped and re-armed

FIGURE 15. Keep Alive Timer with PINGREQ

5.4.2 Will messages

A Will Message arises in a case where a Client is unexpectedly disconnected. When the client is disconnected, applications depending on the client do not receive any notification of the client demise. However, the client can specify a Will Message along with a Will QoS and Will RETAIN Flag in the CONNECT message pay load. Therefore, if the client unexpectedly disconnects, the server sends the Will Message on behalf of the client to all subscribers. Figure 17 shows the Client-Server-subscriber Will Messaging.

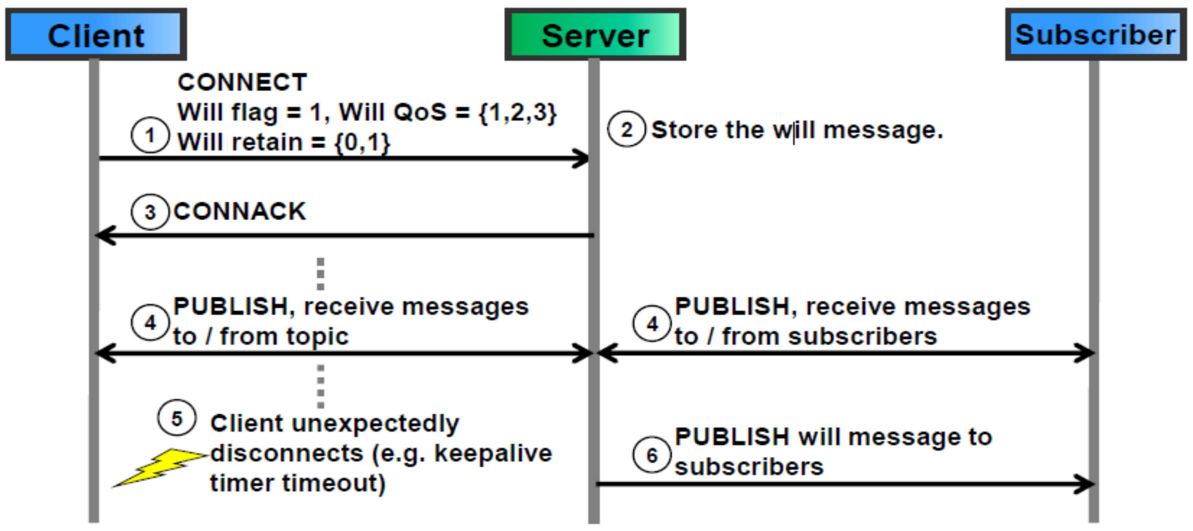

Client
Server
Subscriber
CONNECT
Will flag = 1, Will QoS = {1,2,3}
Will retain = {0,1}
1
2
Store the will message.
3
CONNACK
4
PUBLISH, receive messages to / from topic
4
PUBLISH, receive messages to / from subscribers
5
Client unexpectedly disconnects (e.g. keepalive timer timeout)
6
PUBLISH will message to subscribers

FIGURE 16. MQTT Will Messaging between Client-Server-subscriber.

5.4.3 Topic wildcards

Subscribers are often interested in a great number of topics. However, subscribing to every named topic is time and resource consuming. Therefore, MQTT Topic Wildcard is used when a client wants to receive messages of different topics with a similar structure at once. Topics can be organized through the wildcards path-type topic strings and the Wildcard characters; the forwards slash (/), the number sign (#) and the plus sign (+). Table 15 describes the Wildcard characters and their meaning.

TABLE 15. MQTT Wildcard characters and their meaning

Wildcard	Symbol and example	Meaning
Topic Level Separator	/ my/thesis/topic	It is used to separate each level within a topic tree and provide a hierarchical structure to the topic space.
Single-level Wildcard	+ my/+/topic	It matches one complete topic level. It can be used more than once in a topic subscription.
Multi-level Wildcard	# my/#	It matches multiple topic levels. It must be the last character of a topic subscription.

5.5 MQTT security

Security is prominent no matter whether it concerns bank transfers, online shopping or accessing personal documents over the Internet. Moreover, the main idea behind the IoT technology is to connect every object, such as cars, home and industrial machines, so that it is possible to efficiently improve processes, either business or personal activities. However, connecting these objects to the Internet means exposing vital and sensitive data over the Internet. Some vital and sensitive information might not be meant for the public consumption and leaking of such data most often damages the reputation of the affected company or person. Hence, there is the need to protect such data from leaking.

The Security in MQTT is divided into multiple layers and each layer prevents different kinds of attacks. The layers at which some levels of security are implemented are the Network level, Transport level and the Application Level.

Implementing communication between a Broker and a client over a secure network or Virtual Personal Network (VPN) is one sure way of providing security to MQTT connections. The best practice for the network level security is a gateway implementation where devices are connected via a gateway and the Broker connected over a VPN. The main role for the gateway is to process and relay information between devices and the Internet.

The Transport Level Security (TLS), a successor of the Secure Sockets Layer (SSL), is a cryptographic protocol designed to provide secure communication between the client and the server over unsecure network and when confidentiality of the system needs to be provided. It operates on top of the TCP providing a secure transport for upper layer protocols, such as HTTP. TLS is a very secure method for encrypting traffic but it is also resource intensive due to its required handshake and an increased Packet overhead. However, since MQTT is built on top of the TCP, it can use TLS to secure traffic between the MQTT client and server. But since TLS is resource intensive and MQTT clients are lightweight and energy is of high priority, encrypting just the payload is sufficient instead of encrypting all the Packet.

According to IANA.org port assignment and standards (Touch & Eliot Lear, Service Name and Transport Protocol Port Number Registry, 2017), MQTT uses or is assigned to port 8883 on the Broker side when TLS is used. It is a standard for MQTT connections when it is used on top of the TLS. Moreover, when MQTT is used over a Plaintext TCP connection, it uses the port 1883 (Touch & Eliot Lear, Service Name and Transport Protocol Port Number Registry, 2017). The TLS and TCP port connections can be mixed and not all clients have to be connected in the same way to the Broker. This means that a client may connect to the Broker with TLS and to another one with the Plaintext over TCP. TLS is a complex protocol. It is resource intensive and computationally expensive thus some target platforms may not support it. But with MQTT it is possible to implement security and secure packets at the Application Layer.

When security is applied at the Application Layer, it is implemented at the data payload where application data resides. The communication between the client and server is ensured so that it is encrypted and the identity is authenticated. The client identifier, username and password credentials can also be used to secure and authenticate devices on the Application Layer. They can secure information transmission with fully implemented transport encryption.

6 COMPARISON OF MQTT AND COAP PROTOCOLS

The Internet of Things network is a complex one due to the large number of physical interconnected IoT devices and the constrained nature of the devices, environment and the lossy type of Network they operate. One of the key challenges of implementing an IoT project is to efficiently support machine-to-machine (M2M) communication in constrained conditions. So far in this work it has been elaborated that MQTT and CoAP are the most promising protocols that can be implemented in those constrained conditions.

Although, MQTT and CoAP are totally different protocols, they have some similarities, such as that they are designed to be used in lightweight devices and in constrained environments. This means that they both work well with Low-Power and constrained network devices. Due to their similarities, choosing the appropriate protocol for the development of an IoT application could be difficult depending on the application. However, there are many factors to be considered while planning the right protocol to be used. In this chapter, a comparison of the MQTT protocol and CoAP protocol will be examined based on performance evaluations from different scenarios done elsewhere.

The main difference between CoAP and MQTT is that CoAP runs on top of the UDP while MQTT runs on top of the TCP. Table 16 illustrates the comparisons:

TABLE 16. Comparison table between MQTT and CoAP

MQTT	CoAP
Mode of communication within MQTT is publish and subscribe that are highly decoupled to each other	CoAP is request and response oriented and has an asynchronous communication model
MQTT generally has larger Packet size. Smaller packets less than 127bytes have a 1byt Packet length find and the maximum Packet size is 256MB	CoAP has smaller Packet size as MTU 1280 bytes for IPv6, 127 bytes for 6LOWPAN and 127 bytes for IEEE 802.15.4
MQTT Header field is 2bytes	CoAP Header field is 4bytes
MQTT allows 16 different messaging types	CoAP allows for 4 messaging types
MQTT supports asynchronous messaging	CoAP support both synchronous and asynchronous messaging
MQTT has 3 levels of application reliability which are the QoS levels	CoAP have 2 levels of application reliability in the form Confirmable (CON) and Non-Confirmable (NON)
Transmitting cycle within MQTT is much slower	CoAP has faster transmit cycle
MQTT is not a RESTful protocol	CoAP is RESTful protocol
MQTT works on flexible topic subscription	CoAP has stable resource discovery mechanism
For security, MQTT is unencrypted but it uses TCP's TLS/SSL security encryption	CoAP uses UDP's DTLS security

6.1 Interoperability within IoT

In general terms, interoperability is the extent by which two or more implemented systems from different manufacturers can connect, speak, share, innovate, operate and use data from each other by relying on each other's services as specified by a common protocol and standard. As stated earlier in this thesis, IoT application areas, such as a smart grid, smart appliances, wearable and fitness devices and health, are the main application domains but they are of different architecture and data models. The main idea for the IoT is to connect any device to the Internet that would be able to connect to any other device(s) or system to be able to exchange data and information. However, the infrastructure of the various IoT application domains lacks interconnectivity methods that could allow the interoperability between, for example the network layers and Application Layers.

Similar to the traditional Internet, the interconnectivity or interoperability of IoT devices and systems happens in varying degrees and at different layers within the communication protocol stack. The layers where interoperability takes place, are the Network layer, Application Layer and Data annotation level. Figure 18 shows the IoT Network layer protocols interoperability architecture with various Low-Power networking protocols, such as ZigBee, Z-Wave, Bluetooth, NFC, and also traditional networking protocols, such as the Ethernet, WiFi and hardware connections.

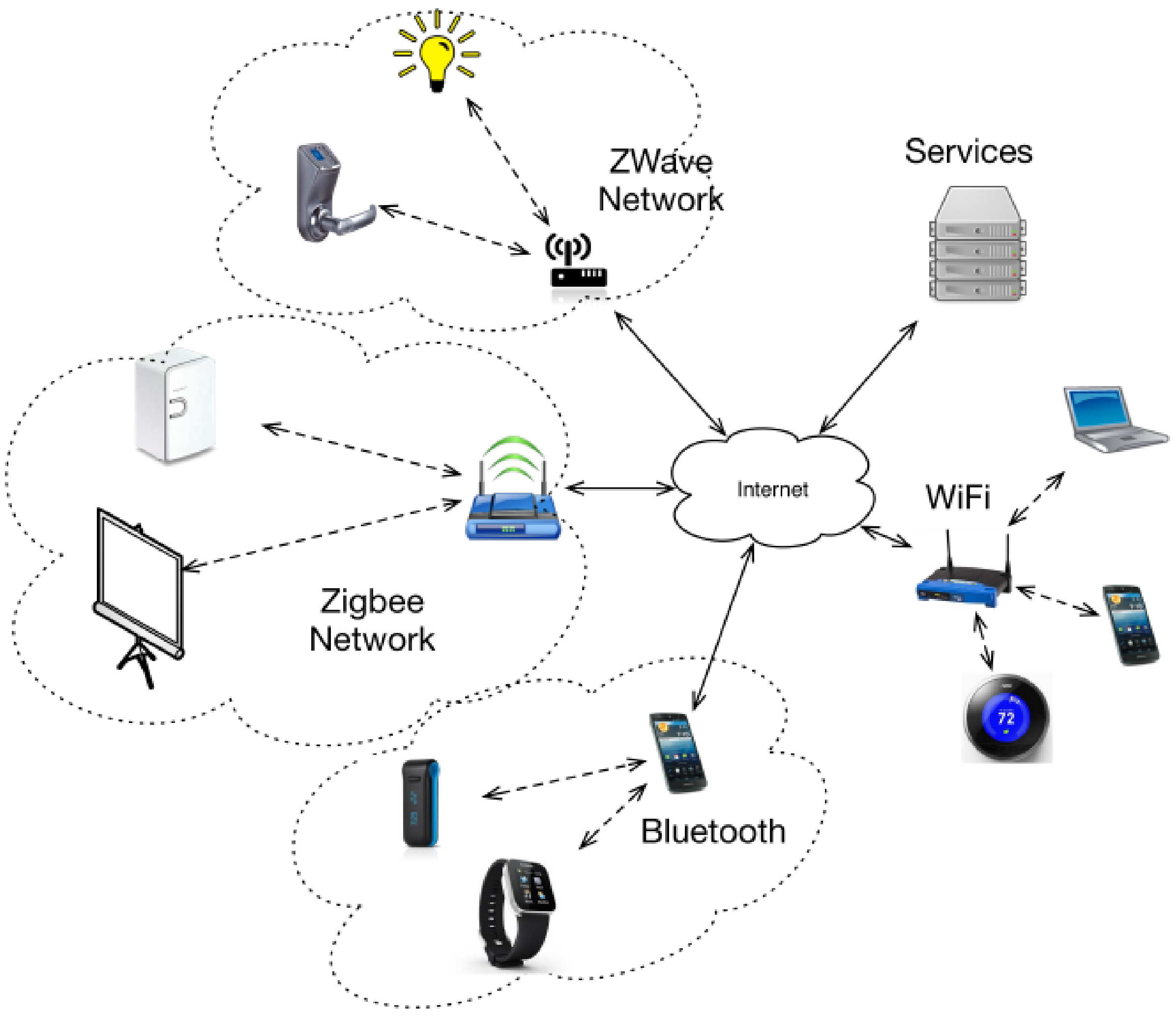


FIGURE 17. IoT Network Layer Interoperability architecture (Pratikkumar, Amit , & Pramod , 2015)

The Network interoperability protocols are designed for a specific domain and applications for some standardized hardware components developed to support multiple networking protocols.

However, this thesis work is on the research of identifying the interoperability among the IoT Application Level protocols specifically between the MQTT and CoAP protocol.

6.1.1 Interoperability between application layer protocols

As elaborated in this thesis, the most competing and proposed Application Level IoT protocols are CoAP and MQTT. Each protocol has a unique characteristics and massaging architecture for an IoT applications. However, the interoperability between devices implemented with these Application Layer protocols and other proposed IoT protocols remains a challenge.

Moreover, there are various interoperability initiatives emerging and working to solve the interoperability challenge within the IoT Application Layer protocols. Most of the initiatives are open source and are built on top of the IoT Application Layer protocols. They focus on the data structure, communication model and semantics of IoT data (OCF Solving The IOT Standards Gap, n.d.).

Notable initiative consortiums are the AllSeen Alliance and AllJoyn (AllSeen Alliance, n.d.) which are open source, universal, secure and development connectivity frameworks with the aim to support and enable the interoperability between the IoT devices. The AllJoyn framework supports device discovery to interoperate and interact. This means that products, applications and services implemented with the AllJoyn framework can connect even without the Internet access to various network layers, regardless of manufacturer or operating system.

Another development framework which has been developed by the Open Interconnect Consortium (OIC) called (IoTivity, n.d.) IoTivity, is also implemented to improve the interoperability between the IoT devices. IoTivity is an open source framework that has discovery of devices mechanisms, data transmission in the form of messaging and streaming model, information exchange and control mechanism, data management, storage, data analyzes from other sources and device management. It also provides a device diagnosis.

6.1.2 Semantics interoperability

Interoperability cannot only be achieved by transferring data with a common data format within the IoT Application Layer protocol. The Semantics interoperability provides a different dimension to the data interoperability at the Application Layer at a higher level than raw data transferred using AllJoyn or IoTivity. The Semantic interoperability means that two separate systems automatically interpret the meaning of data transmitted by the two systems and arrive at same meanings. In terms of IoT platforms, the Sensor Semantic Network (SSN) provides a set of Ontologies and SenML (Sensor Markup Language) also provides Metamodels that are designed to provide interoperability with the application of languages, such as JavaScript Object Notation (JSON), Eclipse Vorto or Eclipse Ponte and Eclipse Franca (Eclipse Foundation Open Source Project Hierarchy, n.d.). Metamodels and Ontologies are related but metamodel is referred as strict set of rules while ontologies are vocabularies (Tayur & Suchithra , 2017).

However, the Application Layer in terms of data methods is divided into sub-layers: Data transfer and semantics. Table 17 shows the IoT Application Layers presented in this sub-layer of the thesis.

TABLE 17. IoT Application Layer with sub-layer

Semantics Layer		Ontology	Metamodel
Data Transfer Layer	Serialization framework	HyperCat	Franca
	IoT-A	AllJoyn/IoTivity	LwM2M
IoT Protocols	HTTP	CoAP	MQTT

The Internet of Things–Architecture (IoT-A), Data Serialization framework, AllJoyn/IoTivity, HyperCat, Ontology, Lightweight Machine-to-Machine (LwM2M), Franca and Metamodel are initiated semantics standard architectures, frameworks and languages proposed by IoT players and partners to have a common ground for the interoperability within the IoT.

The IoT-A project was initiated and proposed by the EU as an IoT architecture model that could allow developers to choose the architecture that will best fit the devices they develop. Data serialization frameworks are open source and they were developed to assist developers to define data and also to enable them to use the data in their preferred programming language. Franca is also a data framework developed by the Eclipse projects. It defines and transforms software interfaces and integrates software components. HyperCat is also an interoperability layer semantic that allows applications to explore data and available resources and also to help to find right URIs. The Lightweight M2M semantic device management protocol designed for sensor networks is suitable for IoT applications that have a low bandwidth and Lossy Networks. LwM2M is developed based on the CoAP and Datagram Transport Layer, bond to UDP and standardized by Open Mobile Alliance (Open Mobile Alliance, n.d.).

7 IOT IN 5TH GENERATION MOBILE COMMUNICATION (5G)

Since the emergence of the IoT, it has gone through various stages of ubiquitous computing with applications built with various types of sensors. As the applications of connected "things", the IoT is expected to grow to an average of 6-7 devices per person by 2020 and with most of the challenges at the device and protocol levels being solved since the past decade. The trend and the challenges currently confronted with the implementation if the IoT is on the integration and interoperability of IoT based systems and other network systems together with mobile data and wireless broadband communication services. Another challenge arises with the cloud computing that requires a new network with the capacity that can handle everything on cloud.

However, the vision of 2020 and beyond (ITU, n.d.) cannot be fully achieved during the current evolution of International Mobile Telecommunications-Advanced (IMT-Advanced) technologies (Blust, 2017) based on the requirements. Hence, the birth of the 5th Generation Mobile Communication Technology (5G), which has been projected to have features over the legacy technologies, will represent the convergence of all Network access technologies. The 5G technology is still in its initial stages and it is yet to be standardized but it has been proposed that the architecture should integrate the need for IoT applications and other seamless integrations. An IoT integration will help in managing the challenges within the IoT networks. This means that there will be fast and high capacity networks for IoT applications, such as a D2D (Devices-to-Device) connection which is expected to form the major network portion of the 5G technology.

Moreover, the economic and social impact of 5G has been reviewed by a research commissioned by Qualcomm technologies (Karen , et al., 2017). As 5G is new and more devices will be connected on 5G, it is expected to generate up to **$12.3 trillion** worth of goods and services of the global economic output in 2035. This according to the research represents or is equivalent to the spending power of US in 2016 on consumer products and it is also more than the consumer spending of China, Japan, Germany, United Kingdom and France combined in 2016. Again, the 5G technology will support up to **22 million jobs** and generate **$3.5 trillion** with the value chain in

2035. This value chain according to the research is approximately the combined revenue of the top 13 companies on the 2016 Fortune Global 500 list (Fortune Global 500 lists, 2017). Qualcomm research also reports that the 5G development will sustain the global Gross Domestic Product (GDP) growth for a longer term. It has been predicted that the total global GDP will grow from 2020 to 2035 to an equivalent of an economy of the size of India, which is the seventh largest economy in the world at the moment.

7.1 5G technology vision

The vision of the 5G technology has been categorized in three main components: Services, Technology and Standards.

7.1.1 Services

The main objective of the service vision is to connect everything to the cloud. However, to make this connecting everything to the cloud a reality, a major transformation within the mobile technology will take place to deliver the much needed ubiquity, low latency and adaptability to transform the entire industry. This transformation is the core of the evolution of the 5G technology and efforts are being made to enhance the Mobile Broadband termed as eMBB (enhance Mobile Broadband) Network as one aspect to realize the 5G vision. The Mobile Broadband enhancement will improve the network and enable efficient data transmission. The cost per bit for data transmission will be much lower, which will increase the use of the Mobile Broadband Network. Thus, an improvement on the Mobile Broadband Network will support and extend the cellular coverage into a wider range of structures, such as office buildings, industrial environment, shopping malls and large venues.

Another service vision of 5G, and the most important or the main core reason for its birth, is to extend IoTs into a Massive Internet of Things (MIoT). The machine-to-Machine (M2M) IoT application will be improved by the 5G technology and will be termed D2D that will enable a significant increase in the adoption and utilization across all sectors. 5G will improve Low-Power requirements and will have the ability to operate both in the licensed and unlicensed spectrum and increase in the coverage area where cost within the MIoT will be much cheaper than what it is today.

IoT is already in existence and many applications are being rolled out operating with older generations of mobile and cellular technologies and other Low-Power wireless technologies operating in both licensed and unlicensed spectrum. However, while waiting for the 5G MIoT to be implemented and rolled out, efforts are being made to improve the current cellular technology Long Term Evolution (LTE) to improve the cellular IoT market. Technologies such as the LTE Cat-M1 enhancedMachine Type Communication (eMTC) and the LTE Cat-NB1 Narrowband IoT (NB-IoT), are being started to incorporate Low-Power to enable a cellular IoT. These LTE IoT cellular network technologies deployments are expected in 2017 after major operators worldwide, such as AT&T (AT&Tnewsroom, 2016), China Telecom (Joseph, 2016), SK Telecom (Agam, 2016), Verizon (Brumfield, 2016) and Vodafone (Ibbetson, 2016) have committed to it. The above-mentioned technology (NB-IoT and eMTC) which will be enabled by the various telecom companies around the world, is a foundation for MIoT which will improve and extend the Low-Power operational capabilities, have an ability to utilize both licensed and unlicensed spectrum and reduce costs due to the economic scale.

Another 5G vision of importance is the Mission Critical Services (MCS) which, when implemented, will support IoT applications, such as industrial automation, remote patient monitoring, smart grid connectivity, autonomous vehicles and commercial drones, that require a high reliability, an ultra-low latency connectivity with a high security and availability. Figure 19 illustrates the 5G vision.

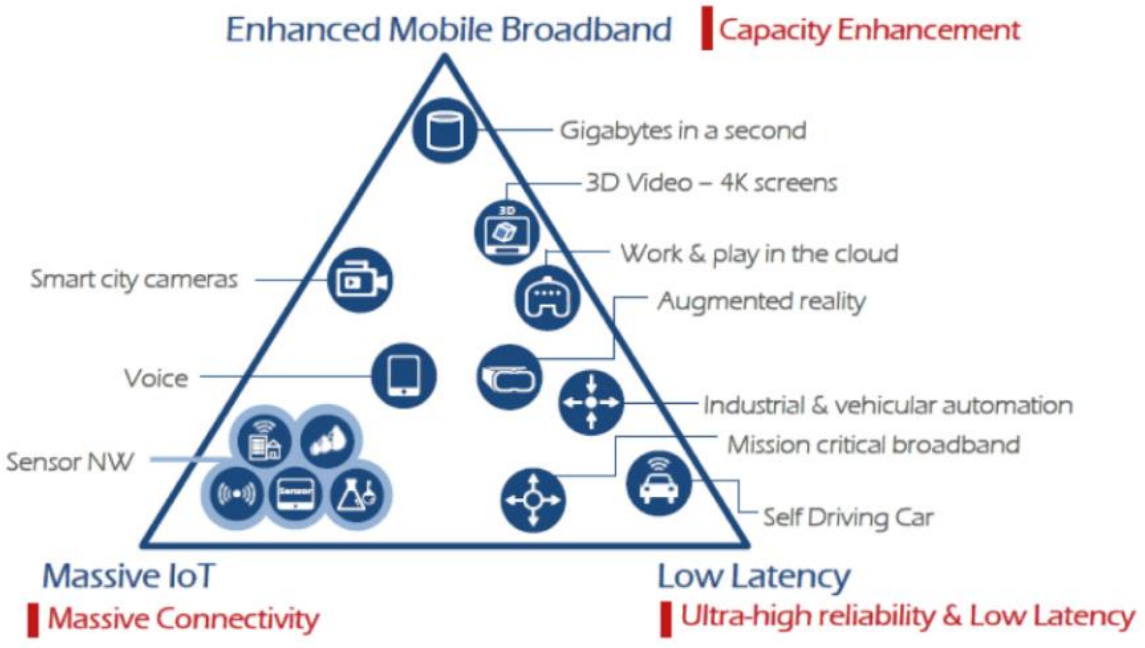


FIGURE 18. 5G vision and usage scenarios for 2020 and beyond (Mallinson, 2016)

7.1.2 5G target performance

The 5G target technical performance requirements have been defined by ITU-R-IMT-2020 (SG05, 2017) for the purpose of consistency in definitions, specifications and evaluations to ensure that manufacturers, application developers, network operators, service and content providers and users do not operate below the minimum performance requirements. This means that any interested group working on the 5G technology must fulfil these minimum requirements for the work to be considered by ITU-R for IMT-2020.

However, these minimum requirements do not restrict the full range of capabilities or the performance Radio Interface Technologies (RITs) might have. It gives room for further and advanced performance in order to achieve IMT-2020. Table 18 is a summary of the ITU-R for IMT-2020 minimum technical performance requirements.

TABLE 18. ITU-R for IMT-2020 minimum technical performance requirements (SG05, 2017)

Metric	Performance Requirement	Definition
Peak Data Rate	DownLink (DL) is 20Gbit/s UpLink (UL) is 10Gbit/s	It is the received Data bits rate under ideal conditions by a single eMBB mobile station assuming all assignable radio resources are utilized
Peak Spectral Efficiency	DL is 30bit/s/Hz (assuming 8 streams) UL is 15bit/s/Hz (assuming 4 streams)	It is the maximum received Data bit rate under ideal conditions by a single eMBB mobile station assuming all assignable radio resources are utilized
User Experience Data Rate	DL is 100Mbits/s UL is 50Mbits/s	It is 5% point of the Cumulative Distribution Function (CDF) of the eMBB user throughput. That is the number of bits correctly received by the user during the active period
5th percentile user spectral efficiency	Reference to TABLE 19	It is the 5%-point CDF of the normalized user throughput.
Average spectral efficiency	Reference to TABLE 20	It is the average throughput of all users corresponding to the number of correctly received bits in the eMBB
Average Traffic Capacity	DL is 10Mbit/s/m^2 in the Indoor Hotspot-eMBB	It is the total traffic throughput served per geographical area. That is the correctly received bits per an area
Latency		
User Plane Latency	4ms for eMBB 1ms for URLLC	Single user for small IP packets for both DL and UL eMBB and URLLC (Ultra-Reliable and Low Latency Communications)
Control Plane Latency	20ms (encouraged to consider lower control latency 10ms)	Transition from Idle to

		Active (eMBB and URLLC)
Connection Density	1000 000 devices per km^2	For mMTC (massive Machine Type Communications)
Energy Efficiency	Efficient Data transmission in a loaded case Low energy consumption when there is no Data	Evaluation in the eMBB scenario
Reliability	99.9999% (1-10^{-5}) success probability	Evaluation in the URLLC scenario for 32 bytes in layer 2 within 1ms at cell edge
Mobility	Stationary: 0km/h **Pedestrian:** 0km/h - 10km/h **Vehicular:**10km/h-120km/h High speed vehicular: 120km/h - 500km/h	Evaluation in the eMBB scenario
Mobility Interruption Time	0ms	Evaluation in the eMBB and URLLC scenarios
Bandwidth	At least 100MHz and up to 1 GHz for above 6GHz operations	Evaluation in the eMBB and URLLC scenarios

TABLE 19. 5th percentile user spectral efficiency performance (SG05, 2017)

Test environment	DL (bit/s/Hz)	UL (bit/s/Hz)
Indoor Hotspot-eMBB	0.3	0.2
Dense Urban-eMBB	0.225	0.15
Rural-eMBB	0.12	0.045

TABLE 20. Average spectral efficiency performance (SG05, 2017)

Test environment	DL (bit/s/Hz/TRxP)	UL (bit/s/Hz/TRxP)
Indoor Hotspot-eMBB	9	6.75
Dense Urban-eMBB	7.8	5.4
Rural-eMBB	3.3	1,6

As mentioned earlier in this chapter, one of the main targets of the 5G technology is to massively connect everything to realize the full roll out of the IoT. However, it can also be deduced from the 5G target performance in table 18 that it is really gearing towards the MIoT. A capacity of 1,000 to 5,000 more than the capacity of 3G and 4G networks will be delivered and it will support cells peak rates between 20Gbit/s and 10 Gbit/s. With the high capacity and peak rate, a connection density of about one million (1M) devices within one-kilometer (1km) area could be achieved. Furthermore, an energy efficiency monitor is expected to be implemented to monitor an efficient energy consumption during the data transmission and to give a very low energy consumption when devices or sensors are idle. It is also targeted to perform on ultra-low latency of about 1-10 milliseconds (1-10ms) of data transmission from one point to another, compared to the 40-60 milliseconds of today’s 3G and 4G Networks. This target performance will support applications, such as fast-moving vehicles at speeds 120km/h-500km/h, where the delivery of information or data between the source and the destination will be within five milliseconds.

Another goal of the 5G performance is the interoperability between 5G, 4G and WiFi, in which a separation of commutations infrastructures will be done to allow mobile users move freely between these infrastructures without any break in connection. This means that for example, cellular networks will be integrated with other communication infrastructures, such as WiFi, and a user will not experience a connection break when moving between the networks. Furthermore, another performance feature will be that the networks will become programmable. This means that operators will be able to make changes to the network to best suit, for example, its customer needs without the need to touch the physical infrastructure. This will be a reality when the 5G Advanced Network infrastructure is implemented using the Software-Defined Network (SDN) and the Network Functions Virtualization (NFV).

7.1.2.1 5G standardization plans

As stated earlier, the 5G technology is at its initial stage and still needs to be standardized. The standardization is based on the International Telecommunication Union (ITU) timeline and the key technologies involved are as shown in the figure 20.

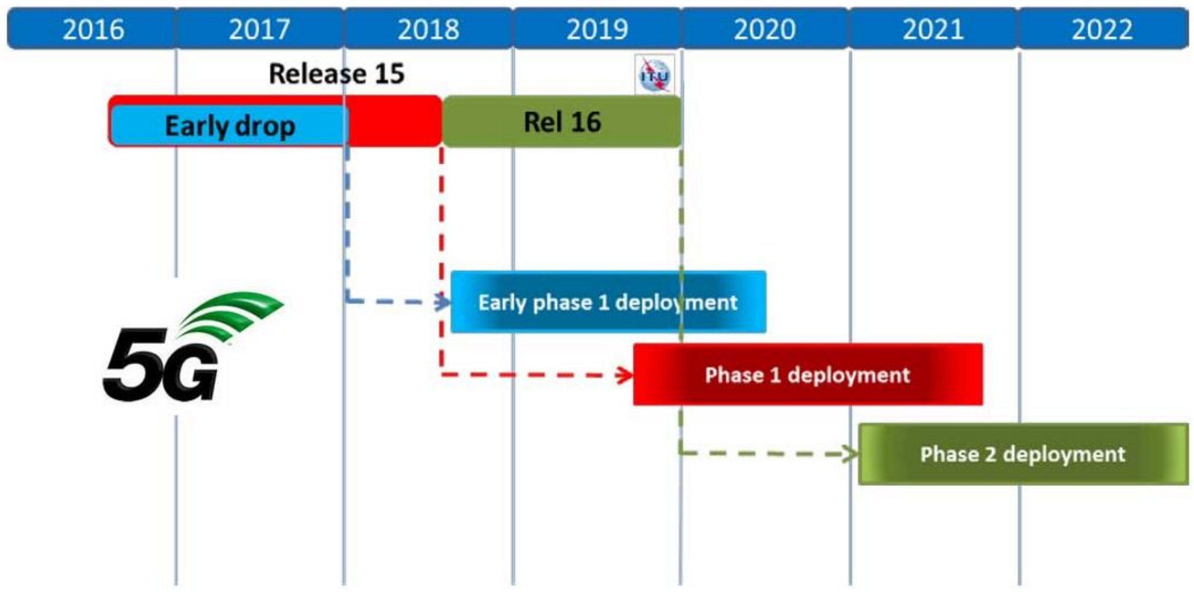


FIGURE 19. 5G technology standardization timeline (Romano, 2017)

3GPP has categorized into a phase based approach and each phase comprises one or more study items and one or more work items. These phases are also termed as release standards where the release 13, which is based on the existing LTE-A, and the release 14 marked the beginning of the study into the 5G technology are already standardized. The Release 15, also termed phase one (1), is the beginning of the 5G standardization and it is aiming at enabling the phase 1, which is expected to be deployed in 2020. The Release 16 will help users into further enhancements and is ready for the products that will make up the 5G technology.

Another form of the standardization process as presented at the IMT-2020 workshop in Munich, Germany is as shown in figure 21 and figure 22 below.

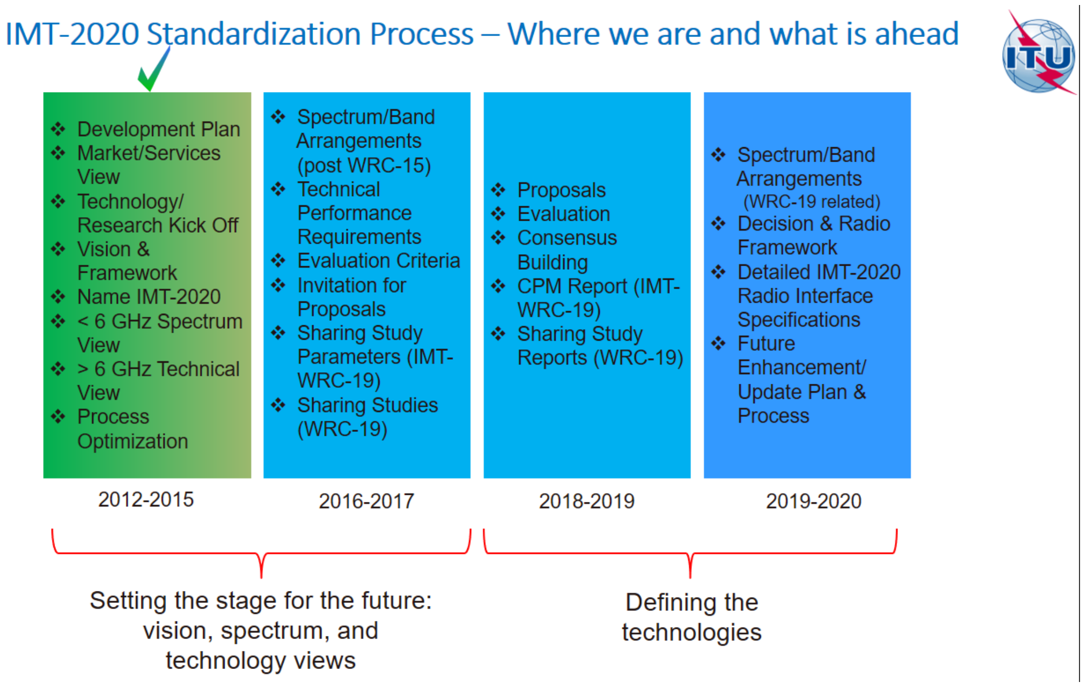

IMT-2020 Standardization Process – Where we are and what is ahead
Development Plan
Market/Services View
Technology/ Research Kick Off
Vision & Framework
Name IMT-2020
< 6 GHz Spectrum View
> 6 GHz Technical View
Process Optimization
Spectrum/Band Arrangements (post WRC-15)
Technical Performance Requirements
Evaluation Criteria
Invitation for Proposals
Sharing Study Parameters (IMT-WRC-19)
Sharing Studies (WRC-19)
Proposals
Evaluation
Consensus Building
CPM Report (IMT-WRC-19)
Sharing Study Reports (WRC-19)
Spectrum/Band Arrangements (WRC-19 related)
Decision & Radio Framework
Detailed IMT-2020 Radio Interface Specifications
Future Enhancement/ Update Plan & Process
2012-2015
2016-2017
2018-2019
2019-2020
Setting the stage for the future: vision, spectrum, and technology views
Defining the technologies

FIGURE 20. IMT-2020 Standardization process (Ying Peng, 2017)

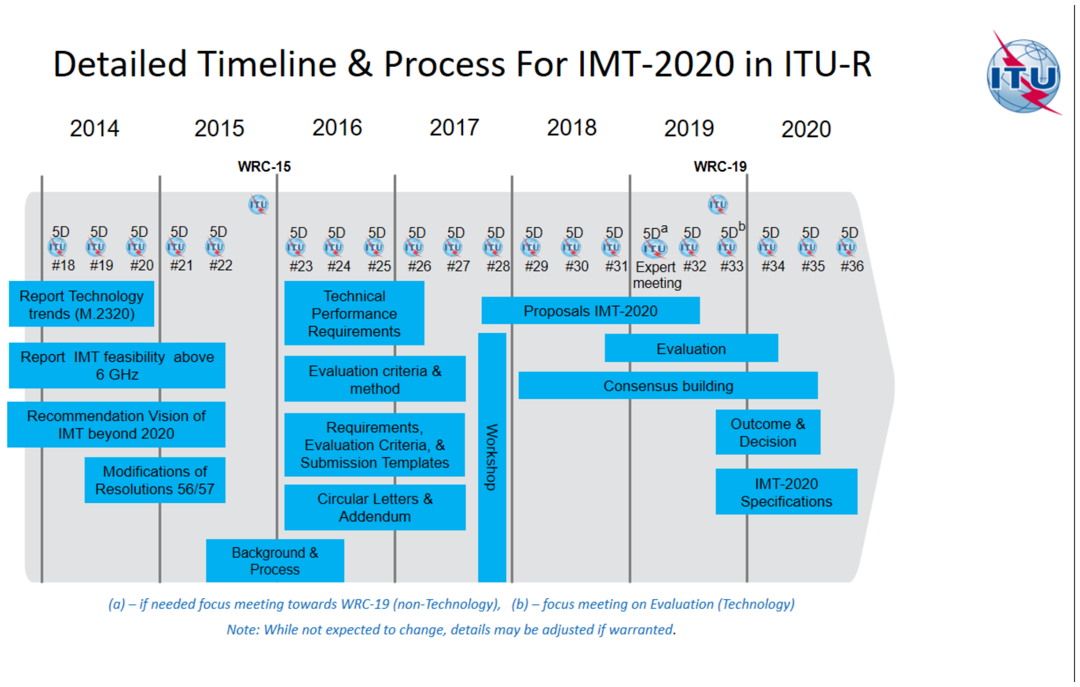

Detailed Timeline & Process For IMT-2020 in ITU-R
2014
2015
2016
2017
2018
2019
2020
WRC-15
WRC-19
5D #18
5D #19
5D #20
5D #21
5D #22
5D #23
5D #24
5D #25
5D #26
5D #27
5D #28
5D #29
5D #30
5D #31
5D a Expert meeting
5D #32
5D b #33
5D #34
5D #35
5D #36
Report Technology trends (M.2320)
Report IMT feasibility above 6 GHz
Recommendation Vision of IMT beyond 2020
Modifications of Resolutions 56/57
Background & Process
Technical Performance Requirements
Evaluation criteria & method
Requirements, Evaluation Criteria, & Submission Templates
Circular Letters & Addendum
Workshop
Proposals IMT-2020
Evaluation
Consensus building
Outcome & Decision
IMT-2020 Specifications
(a) – if needed focus meeting towards WRC-19 (non-Technology), (b) – focus meeting on Evaluation (Technology)
Note: While not expected to change, details may be adjusted if warranted.

FIGURE 21. Detailed timeline and process for IMT-2020 in ITU-R (Ying Peng, 2017)

7.1.3 5G key technologies

As the standardization of the 5G is ongoing, there are some key technologies that are being considered and to be enabled. These are the Advance Network Millimeter Wave (mmWave) system, Multi-Radio Access, Advanced Massive Multiple Input Multiple Output (MIMO), Multiple access, advance Device-to-Device (D2D), and an advanced small cell.

With the Advanced Network technology, the aim is to create an integrated and distributed network function which is programmable using the network software, such as SDN and NFV. With SDN, the network control can be programmed to allow flexibility of enhancing the network features and to aid in data forwarding paths and functions. The NFV is a technology used to virtualize a complex hardware based network node function into software building blocks that can be combined a chained to create advanced communication service (Chung, 2017). It eliminates dependency and complex hardware based Network nodes using flexible software blocks that are called Virtualized Network Functions (VNTs).

The millimeter Wave system has huge bandwidth in the mmWave band, which is has a frequency above 6 GHz more than LTE mmWave band. More capacity can be gained with mmWave, for example 2.2Gbits/s of data rate can be supported by the 28GHz band using a multi-cell and 500MHz bandwidth. The Massive MIMO on the other hand will enhance a data rate using the Full-Dimension MIMO (FD-MIMO), the spectral efficiency will be enhanced using Multi-User MIMO and the Energy efficiency and data rate will be enhanced accordingly using Virtual MIMO (MIMO). The advanced D2D technology proposed for the 5G is critical for the IoT. With Advanced D2D, offloading data from a mobile network so that the loading and cost of processing data and signaling is reduced. Mission Critical Push-to Talk (MCPTT) is another technology emanating from the Advanced D2D that will support the Vehicle-to-Anything (V2X) communication. A throughput enhancement could be achieved significantly by the Small Cells technology where a large number of small cells in a given area will be used to realize this. Small cells are easy to deploy, self-configure and distribute.

8 CONCLUSION

This thesis work was started by studying a little bit of the history of the IoT and the global economic impact on the world market at large. It has been reviewed that by 2020, about 26 billion units, excluding computers and smartphones, will be connected to the Internet. Having many devices connected will in return generate some revenue in the global economy of about $1.9 trillion through sales and other markets by 2020.

Furthermore, the existing IoT architecture, and standards that enable protocols were presented. Chapter 2 deals with the IoT architecture where the IoT reference model is compared with the traditional Internet to identify the differences and similarities and the applications are also reviewed at each level. A review on the IoT gateway connectivity protocols and the IoT protocol stacks, which provide the end to end connectivity from a sensor to the backend application, were discussed in Chapter 0. Here, the important features and functionalities of IoT gateway protocols were reviewed and the IoT Application Level protocols, such as HTTP and RESTful, were discussed. It was reviewed that even though HTTP is widely used on the www and has been standardized, it is not suitable for many IoT applications due to some limitation on constrained devices.

Then, Application Layer communication protocols, CoAP and MQTT were carefully and extensively inspected in detail in chapters 4 and 5. These IoT Application Layer protocols have been tipped as the most suitable protocols currently being implemented in most IoT applications, the reason being that they are light weight and are suitable for constrained devices, such as sensors. Chapter 6 is dedicated to the comparison and interoperability between the two main IoT protocols, MQTT and CoAP. In comparison, it was reviewed that it all depends on the preference and the application. Either protocol is suitable for IoT applications because both are designed for lightweight devices and suitable to a constrained environment. On the question on the interoperability, it was reviewed that applications implemented with either MQTT and CoAP protocol will not just interoperate even though they are similar but each has unique characteristics and messaging architecture. Therefore, to achieve the interoperability between any IoT protocols, a semantics interoperability must be applied. The Semantic interoperability provides a different dimension to the data interoperability at the Application Layers protocols. This means that interoperability must take place at a higher level of the protocol stack than raw data transferred.

Finally, Chapter 7 was dedicated to review the future of the IoT and the cellular communication technologies. It was reviewed that one of the major objectives of the much talked cellular technology; the 5G technology, is to massively connect “things”. Therefore, one of the core technologies being implemented is the MIoT. It is expected that the IoT will grow to an average of 6-7 devices per person by 2020 and with most of the challenges at the device and protocol levels being solved. That interconnectivity and interoperability between devices and protocols will have the same level of operation and will eliminate the interoperability challenges facing the current IoT applications implementation.

Even though, 5G is still at its initial stages and it is yet to be standardized and deployed by the year 2020, the IoT is included in the plan of 5G where input/output or sensors / actuators, IoT platforms and positioning are planned to be integrated. After 2020 we are going to experience about 26 billion units or devices being connected to the Internet.

9 REFERENCES

1. (ISO), I. O., & Iec, I.1994. Information Technology-Open Syatems Interconnection-Basic Reference Nodel: The Basic Model. Switzerland: Iso/Iec.
2. Agam, S. 2016, July 5. Sk Telecom's Now Offering Smartphone-Like IoT Data Plans On Its New Nationwide Wireless Network In South Korea. Network World. Date of retrieval15.9.2017. https://www.networkworld.com/article/3090893/like-smartphones-iot-communications-going-long-distance.html
3. Allseen Alliance. (N.D.). Retrieved From Https://Allseenalliance.Org/Framework
4. Ashton, K. 2009, June 22. That 'Internet Of Thing' Thing. Rfid Journal, 7, 97-114.
5. Asín, A., & Gascón, D. 2016. Libelium Comunicaciones Distribuidas S.L. Libelium World. Date of retrieval15.1.2017. http://www.libelium.com/resources/top_50_iot_sensor_applications_ranking/#show_infographic
6. AT&Tnewsroom. 2016, October 26. At&T Launches North America’s First Lte-M Site to Grow Iot. (AT&T). Date of retrieval15.9.2017. http://about.att.com/story/north_americas_first_ltem_site_to_grow_iot.html
7. Blust, S. M. 2017, 09 17. IMT-Advanced Standards for Mobile Broadband Communications. (International Telecommunication Union (ITU-R)) http://www.itu.int/net/newsroom/wrc/2012/features/imt.aspx
8. Bormann, C., Jimenez, J., & Melnikov, A. 2010. Constrained Restful Environments (CoRE). (The Internet Engineering Task Force (IETF)). Date of retrieval 04.03.2017. https://datatracker.ietf.org/wg/core/charter/
9. Bradley, J., Reberger, C., Dixit, A., & Gupta, V. 2013. Value of The Internet of Everything for Cities, States & Countries. Cisco Systems .
10. Brumfield, J. 2016. Verizon To Lead In Deploying Cat M1 Lte Network Technology for IoT. (Verizon). Date of retrieval15.09.2017. http://www.verizon.com/about/news/verizon-lte-network-technology-iot
11. Chase, J. 2013. The Evolution of The Internet of Things. Texas Instruments Incorporated, 1-7.
12. Chen, X. 2014. Constrained Application Protocol for Internet Of Things. 1-12.
13. Chung, J.-M. (2017). Internet Of Things & Augmented Reality Emerging Technologies. (Yonsei University). https://www.coursera.org/learn/iot-augmented-reality-technologies
14. Cohn , R. J., Cophen, R. J., Banks, A., & Gupta, R. 2015. Mqtt V3.1 Protocol Specification. Oasis .
15. Dr.Vermesan, O., Dr. Friess, P., Guillemin, P., Gusmeroli, S., Sundmaeker, H., Dr. Bassi, A., . . . Dr. Doody, P. 2011. Internet of Things Strategic Research Roadmap. IoT Cluster Strategic Research Agenda 2011, 9-52.
16. Eclipse Foundation Open Source Project Hierarchy. (N.D.). (Eclipse). 2016. Date of retrival 05.08.2017. http://www.eclipse.org/projects/tools/hierarchy.php
17. Evans, D. 2012. The Internet of Everything: How More Relevant and Valuable Connections Will Change the World. Cisco Internet Business Solutions Group (IBSG), 1-9.

18. Fielding, R., & Reschke, J. 2014. The Internet Engineering Task Force (IETF Tools). Date of retrieval 03.04.2017. https://tools.ietf.org/html/rfc7230#page-5
19. Fortune Global 500 Lists. (2017). (Fortune) Retrieved September 15, 2017, From Http://Fortune.Com/Global500/List/
20. Fremantle, P. 2015. A Reference Architecture for The Internet of Things. London: WS02.
21. Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. 2013. Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. p.1-19.
22. Ibbetson, L. 2016. Enabling the Internet of Things with NB-IoT. Vodafone Group. Date of retrieval 15.09.2017. http://www.vodafone.com/content/index/what/technology-blog/enabling-iot.html
23. Iotivity. N.D. https://www.iotivity.org/
24. Ip, C. 2016. Internet of Things: The IoT Opportunity – Are You Ready to Capture A Once-In-A Lifetime Value Pool? Mckinsey & Company IoT Conference. Hong Kong.
25. ITU. N.D. ITU Towards “IMT for 2020 and Beyond”. (International Telecommunications Union (Itu)). http://www.itu.int/en/itu-r/study-groups/rsg5/rwp5d/imt-2020/pages/default.aspx
26. Joseph, W. 2016. China Telecom Targets Nationwide NB-IoT Coverage Next Year. (Mobile World Live). Date of retrieved 15.09.2017. https://www.mobileworldlive.com/asia/asia-news/china-telecom-targets-nationwide-nb-iot-coverage-next-year/
27. K. H. 2015. Observing Resources in the Constrained Application Protocol (CoAP). (Internet Engineering Task Force (Ietf)). Date of retrieval15.04.2017. https://tools.ietf.org/html/rfc7641
    Karen. C, Jim D., Bob F., BillM., Brendan O., & Francis S. 2017. The 5G Economy. Qualcomm. Date of retrieval 20.09.2017. https://www.qualcomm.com/invention/5g/economy
28. Macgillivray C. 2016. IDC Futurescape: Worldwide Internet of Things 2017 Predictions. IDC Web Conference.
29. Mallinson K. 2016. The Path to 5G: As Much Evolution as Revolution. 3GPP. Date of retrieval 5.10.2017. http://www.3gpp.org/news-events/3gpp-news/1774-5g_wiseharbour.
30. Meulen R. V. 2015. Gartner says 6.4 Billion Connected "Things" Will be in use in 2016, up 30 percent from 2015. Gartner Inc. Barcelona.
31. Mit, E. K. (N.D). World Wide Web Consortium (W3C). date of retrieval 03.04.2017. https://www.w3.org/standards/
32. Obermaier D. (N.D.). Getting started with MQTT a Protocol for the Internet of Things. (Dzone Refcardz). Date of retrieval 05.06.2017. https://dzone.com/refcardz/getting-started-with-mqtt
33. OCF Solving the IoT Standards Gap. (n.d). (Open Connectivity Foundation (OCF)). https://openconnectivity.org/
34. Open Mobile Alliance. (n.d). (Oma). http://openmobilealliance.org/
35. Pratikkumar D., Amit S., & Pramod A. 2015. Semantic Gateway as a Service Architecture for IoT Interoperability. New York: IEEE.
36. Rescorla E., M. N. 2012. Datagram Transport Layer Security Version 1.2. (Internet Engineering Task Force (Ietf)). Date of retrieval 05.05.2017. https://tools.ietf.org/html/rfc6347

37. Richard J Coppen, A. B. 2016. Information Technology - Message Queuing Telemetry Transport (MQTT) V3.1.1. (International Organization For Standardization). Date of Retrieval 05.05.2017. https://www.iso.org/standard/69466.html
38. Richard Möller, S. B. 2016. Ericsson Mobility Report: Growth in the Number of Connected Devices is Driven by Emerging Applications and Business Models, and Supported by Falling Device Costs. Stockholm: Ericsson Mobility Report.
39. Romano, G. 2017. Preparing the Ground for IMT-2020. (3GPP). Date of retrieval 08.10.2007. http://www.3gpp.org/news-events/3gpp-news/1901-imt2020_news
40. SG05, I.-R. 2017. Draft New Report Itu-R M.[IMT-2020.Tech Perf Req] - Minimum Requirements Related to Technical Performance for IMT-2020 Radio Interface(s). (ITU-R IMT-2020).04.10.2017. https://www.itu.int/md/r15-sg05-c-0040/en
41. Shelby Z., Hartke K., & Bormann C. 2014. Date of retrieval 03.04.2017. Internet Engineering Task Force (Ietf): https://tools.ietf.org/html/rfc7252
42. Shelby Z., Hartke K., & Bormann, C. 2014. The Constrained Application Protocol (CoAP). (Internet Engineering Task Force (Ietf)). Date of retrieval 15.04.2017. https://tools.ietf.org/html/rfc7252
43. Shimonski R. 2005. Network+ Study Guide & Practice Exams. Syngress.
44. Tayur V. M., & Suchithra , R. 2017. Review of Interoperability Approaches in Application Layer of Internet of Things. IEEE, 322-326.
45. Touch J., & Eliot Lear A. M. 2017. Service Name and Transport Protocol Port Number Registry. Date of retrieval 12.08.2017. https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml?&page=112
46. Touch J., & Eliot Lear, A. M. 2017. Service Name and Transport Protocol Port Number Registry. Date of retrieval 11.08.2017. https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml?&page=33
47. Vermesan Ovidiu, P. F. 2014. Internet of Things-from Research and Innovation to Market Deployment. River Publishers, 1-143.
48. Wikibooks. 2015. Date of retrieval 3.04.2017 https://en.wikibooks.org/wiki/communication_networks/http_protocol
49. World R. 2016. MQTT Architecture,working Operation,use cases. Rf Wireless World: http://www.rfwireless-world.com/tutorials/mqtt-tutorial.html
50. Ying Peng, J. J. 2017. Guidelines for Evaluation of Radio Interface Technologies for IMT-2020 “Report ITU-R M.[IMT-2020.EVAL]”. Munich: 3GPP.