Master's Thesis - Communication Systems and Networks

Comparison and Evaluation of LwM2M and MQTT in Low-Power Wide-Area Networks

By: **Alessandro Parmigiani**

First Examiner: Prof. Dr.-Ing. Uwe Dettmar (TH Köln)

Second Examiner: Prof. Dr.-Ing. Harald Elders-Boll (TH Köln)

May 2021

Fakultät für
Informations-, Medien-
und Elektrotechnik

Technology
Arts Sciences
TH Köln

Statutory Declaration

I hereby confirm on my honor that this thesis is the result of my independent work. All sources and auxiliary material used in this thesis are cited completely.

Bratislava, 21st May 2021

Alessandro, Parmigiani

Master’s Thesis

Title: Comparison and Evaluation of LwM2M and MQTT in Low-Power Wide-Area Networks

Reviewers:

- Prof. Dr.-Ing. Uwe Dettmar (TH Köln)
- Prof. Dr.-Ing. Harald Elders-Boll (TH Köln)

Abstract: This master’s thesis investigates and compares the performances of LwM2M and MQTT operating in LTE-M and NB-IoT. The experiments consider the traffic-over-the-air generated by the two IoT protocols, delivering information about the packets captured and the number of bytes exchanged. Moreover, the average energy consumption analysis describes how LwM2M and MQTT may influence devices. Also, the work explains how security authentication methods as well as cellular networks’ properties may impact on the protocols’ efficiency. Finally, the results’ evaluation draws a conclusion about the impact of LwM2M of MQTT in cIoT and how these perform compared to each other.

Keywords: LwM2M, MQTT, LTE-M, NB-IoT, cIoT, energy consumption, traffic-over-the-air, LPWAN, IoT

Date: 21st May 2021

Acknowledgements

As this work represents an important step into future challenges, I would like to express my gratitude to everyone who supported me, directly or indirectly. In particular, I would first like to thank my thesis' advisors: Prof. Dr.-Ing. Uwe Dettmar and Prof. Dr.-Ing. Harald Elders-Boll. Not only they supervised this thesis but also guided me through both Bachelor's and Master's studies. Secondly, a warm thanks goes to all my colleagues at 1NCE. Especially, Jan, Shireen, Mohammed and Hatim, who motivated me and helped me throughout this 6-month journey. I would also like to acknowledge Dominika, Jan, Johannes and Jošua as the second readers of this thesis. I am gratefully indebted to them for their very valuable comments on this work. Moreover, especially during a global pandemic, long distance friendships give an enormous encouragement reaching personal goals. Therefore, I would like to deeply thank: Albi, Alex, Andre, Davide, Johannes, Jošua, Kristie, Matteo, Miguelito and Paolino. Finally, I must express my very profound gratitude to my relatives, to my parents and to my girlfriend for providing me with unconditional support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Table of Contents

1 Introduction

Internet of Things (IoT) is a global phenomenon receiving increasing interest from both industry and private users. Due to its growing popularity, IoT solutions need to adapt to various use-case requests. Not only within enterprises but also among private users there are new applications required: smart parking, asset tracking, construction equipment monitoring, smart wearables etc. Because of the scalability and remote locations involved, these examples share three main challenges: limited battery-life, growing traffic-over-the-air and remote connectivity.

The first and second challenges go often along with each other and are influenced by the communication protocols used. The quantity and length of packets involved are impacting on energy-consumption and data-usage. Transport protocols like TCP and UDP, for example, have different approaches regarding data-exchange. The first one is connection-oriented and involves more packets, while the second is connectionless and reduces network congestion. Within IoT, TCP and UDP are mostly abstracted by higher-level messaging protocols that can handle back-end and applications requests. Among these one may find: MQTT, CoAP, MQTT-SN, DDS, AMQP and XMPP. The most popular is MQTT, working on top of TCP. It is supported by both Amazon Web Services (AWS) and Azure IoT [1]. Therefore, the IoT solutions providers "will select MQTT due to the relatively strong ecosystem support that has existed for many years" [2, p. 5]. Moreover, there are other solutions defined as device management protocols, which build on top of the previously mentioned messaging protocols and allow users to maintain device's software, hardware, and configuration. The most representative among this category is Lightweight M2M (LwM2M) which builds on top of CoAP. "As network technologies and infrastructures such as 5G expand, the number of LwM2M-based products will increase, and accordingly, a platform for managing a large number of devices will be highly required" [3, p. 1].

The Remote Connectivity challenge, on the other hand, is about understanding the most suitable access technology when devices cannot rely neither on Personal Area Network (PAN) nor on Local Area Network (LAN) available nearby. The choice is therefore between different cellular IoT (cIoT) technologies. Proprietary Low Power Wide Area Network (LPWAN) standards like LoRaWAN and Sigfox are often not available worldwide and require additional infrastructure. On the other hand, industries standards like NB-IoT and LTE-M are present in more areas and allow a faster network development.

1.1 Motivation

The topic of the master's thesis was conceived in collaboration with 1NCE GmbH, a mobile virtual network operator (MVNO) that has one of its main focuses on IoT connectivity. The company has a tight partnership with Amazon AWS which only supports MQTT in its IoT device management platform. Recently, 1NCE's connectivity solution released a CoAP endpoint. This allows customers to send messages via CoAP, converted to MQTT for further elaboration within AWS IoT. As an additional step, 1NCE decided to start a feasibility analysis to embrace LwM2M which works mainly on top of CoAP. The goal is to understand the real impact on devices' energy consumption and data-usage over cellular links compared to MQTT. Since the company is specialized in cellular connectivity, the study should focus on devices' performances using the most

recent cIoT access technologies provided by 1NCE: LTE-M and NB-IoT. This is accomplished by this master's thesis, representing a study on how LwM2M and MQTT perform in both LTE-M and NB-IoT.

1.2 Task Description

The goal of thesis is to compare and evaluate LwM2M and MQTT when operating over LTE-M and NB-IoT. In order to achieve the intent, the work not only includes a theoretical research about the involved topics, but also a practical comparison containing experiments. Consequently, along with the tools used, the work describes observations acquired from preliminary tests involving network, hardware, and software. These are particularly useful to understand events, predict scenarios or adjust configurations according to the expected behavior when comparing the protocols. Finally, a series of measurements using LTE-M and NB-IoT's cellular links allow to draw the comparison and evaluation of LwM2M and MQTT. The detailed task description contains:

1. Collection and comparison of Information and fundamentals regarding the four main backgrounds: LwM2M, MQTT, LTE-M and NB-IoT
2. Choice of appropriate hardware, software, and measurement tools, i.e. board, extension board, modem, power, and network protocol analyzer, as well as LwM2M and MQTT's clients and servers libraries
3. Preparation of the experimental environment including clients and servers' code adaption, cellular network analysis and hardware's current behavior
4. Adjustment and modifications based on preliminary tests involving the setup prepared in 4.
5. LwM2M and MQTT's packets analysis and device's energy consumption investigation first in LTE-M and later in NB-IoT. The measurements include same hardware and software configurations for both cellular technologies as well as equal typical protocol's event phases for the study: initial connection, single device to server message, steady-state update, and mobility environment
6. Comparison and evaluation of the results

1.3 Related Works

There is a recent and detailed work comparing MQTT and LwM2M (2020): "Comparing the efficiency of LwM2M and MQTT: hands-on test results of two technology clients on a typical IoT device" from the IoT testing firm Machnation [2]. The author focuses on the performance analysis of the two protocols within IoT platforms scenarios: AWS IoT for MQTT and AVSystem for LwM2M. Some of the test sessions and their results are listed below:

- Initial Connection to the Broker/Server: "LwM2M and MQTT required 4213 bytes and 14907 bytes, respectively, with the average LwM2M packet size being 9% smaller than the average MQTT packet size"; overall LwM2M is 72% (in terms of packet delivered) more efficient in this step
- "LwM2M devices transmit 31% less data in a steady state than MQTT devices". It refers to a 10-minute steady-state capture window
- OTA Updates delivery: "MQTT is 11% more efficient than LwM2M over CoAP at delivering data during an over-the-air (OTA) firmware update"
- "MQTT devices consume 33% more energy than LwM2M devices when measured at idle and 1, 30, and 60-second update intervals…". LwM2M consumed

0.572 Wh while MQTT 0.433 Wh. The percentage refers to the average of the energy consumption in the different intervals. Regarding this last point the author finds the reason of such a large consumption gap in the framework used. AWS IoT seems to be faster but with high computational effort. Moreover, the test was performed with a Raspberry Pi 4 which lacks a power optimization. This may have negatively influenced the experiment

Machnation shows data optimization for transmission under LwM2M especially in the initial connection and steady-state. MQTT on the other hand wins in the OTA updates. The power-consumption test is not exhausting since it strictly depends on the IoT framework used as well as the hardware's power optimization. Nevertheless, the work concentrates on the comparison of LwM2M and MQTT within IoT platforms. Also, the experiment does not involve cellular networks. Machnation mentions constrained-network environments, like cIoT, as areas of future research for testing LwM2M.

Another related work is "Performance of TCP and UDP over Narrowband Internet of Things (NB-IoT)" by Wirges and Dettmar [4]. Here the analysis regards an NB-IoT link which is preferred to LTE-M due its missing coverage within Germany in 2019. MQTT works on top of TCP which is compared with MQTT-SN on top of UDP. The experiment shows that 89.60% of the packets sent under MQTT were lost against just 3.74% with MQTT-SN. This happens because TCP is extremely sensitive to delays in the link, while UDP not. The authors state that a high packet loss surely compromises the battery efficiency of devices. Moreover, NB-IoT usually operates with minimal signal requirements which makes the use of UDP even more meaningful. Although the use of MQTT-SN ensures a valid and robust comparison with MQTT, it leaves the test of a device management platform like LwM2M within NB-IoT open. Another interesting extension of the work may involve the use of LTE-M for the comparison of TCP- and UDP-based protocols.

1.4 Thesis Structure

This master's thesis is divided into five main Chapters. The first two are about theoretical fundamentals: the first introduces LwM2M, MQTT, LTE-M and NB-IoT, necessary to comprehend this thesis. The second draws a first comparison between IoT protocols and cellular technologies based only on the previously mentioned fundamentals. The third Chapter provides details about the tools utilized during the experiments. On the other hand, Chapter four explains observations and issues deriving from preliminary tests involving cellular networks as well as the tools introduced in Chapter three. It also includes a detailed description of the experiments flow along with the parameters and configurations used. This section is useful to reproduce the experiments. Finally, Chapter five describes and analyzes the measurements. It is divided into four different typical LwM2M and MQTT cIoT devices' activities: initial connection, single device to server message, steady-state update, and mobility scenario. Every activity has its own dedicated section, consisting of results. These are then evaluated to draw a comparison between LwM2M and MQTT via LTE-M and NB-IoT.

2 Fundamentals

This Chapter introduces the relevant protocols and cellular access technologies that are compared throughout this thesis. It starts with Lightweight M2M followed by MQTT. The second part deals with LTE-M and NB-IoT.

2.1 Lightweight M2M

Lightweight M2M (LwM2M) is a device management protocol created by the Open Mobile Alliance (OMA) SpecWorks. Since OMA includes the most influential mobile operators, LwM2M represents the synthesis of the IoT market-needs nowadays. Software AG indicates it as one of the two “Adopt” Connectivity and Protocols trends just behind OpenAPI. This is a clear suggestion for companies as it will be playing a fundamental role in the future [5]. Although it is not an IETF specification it builds on top of RFCs’ protocols and securities standards. The high expectations regarding LwM2M lie in four main aspects: it is a UDP-based protocol running on top of CoAP; it relies on strong security fundamentals like bootstrapping, provisioning of secure credentials and ACL; it incorporates device management features like object management and data model definitions allowing a full adaptability to every use-case; it supports Low Power WAN (LPWAN) like LTE-M and NB-IoT, also in Non-IP Data Delivery (NIDD) environments.

As an application layer protocol, it is comparable with i.e. Message Queue Telemetry Transport (MQTT) but the communication structure is different. First, MQTT is an IoT messaging protocol while LwM2M an IoT device management protocol. Secondly, data sending and retrieve actions just exist between client and server and do not involve a client-broker-client constellation like in MQTT. Thirdly, here is the server orchestrating the request by writing, reading, and asking for resources while the client acts as a passive entity. Nevertheless, LwM2M general use-cases remain the same as other messaging protocols: smart mining, cities, parking, utilities etc. The difference is the condition in which devices may operate: low-bandwidth and low coverage are some of environments where LwM2M or other UDP-based protocols may functionate better than others. As of 2021 relevant IoT cloud platforms like AWS and Azure are not fully supporting LwM2M but rather rely on third-party bridges to access the internal data transfer logic. OMA SpecWorks organizes the LwM2M specification in two parts: the core technical document describes the messaging layer while the transport part how this interfaces to the selected transport method.

2.1.1 Protocol Stack

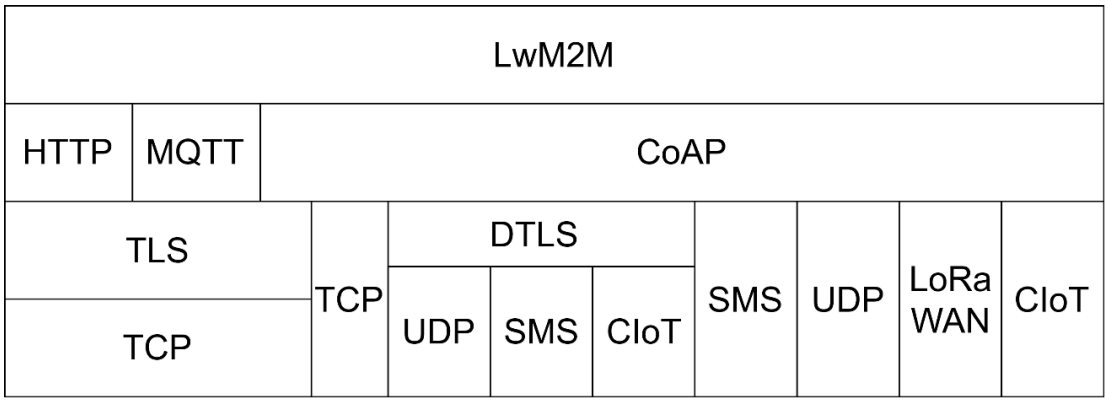

Figure 2.1: LwM2M Protocol Stack [6, p. 17]

LwM2M was developed as a device management protocol on top of CoAP which mainly operates over DTLS and UDP. First in version 1.1 and then in the latest 1.2, OMA expanded LwM2M's protocol stack by including CoAP over TLS, TCP, LoRaWAN and Cellular IoT as well as on top of HTTP and MQTT over TLS and TCP. Figure 2.1 shows the protocol stack of version 1.2.

2.1.2 Lightweight M2M Interfaces

LwM2M interfaces describe how the client and the servers interact with each other. There is a bootstrap server and a regular server involved in the LwM2M enabler. The first one is relevant in the provisioning and security part, while the second one becomes important during the data transfer and in the RESTful request/response model, based on CoAP [7]. The four interfaces are the following:

- Bootstrap Interface
- Registration Interface
- Device Management and Service Enablement Interface
- Information Reporting Interface

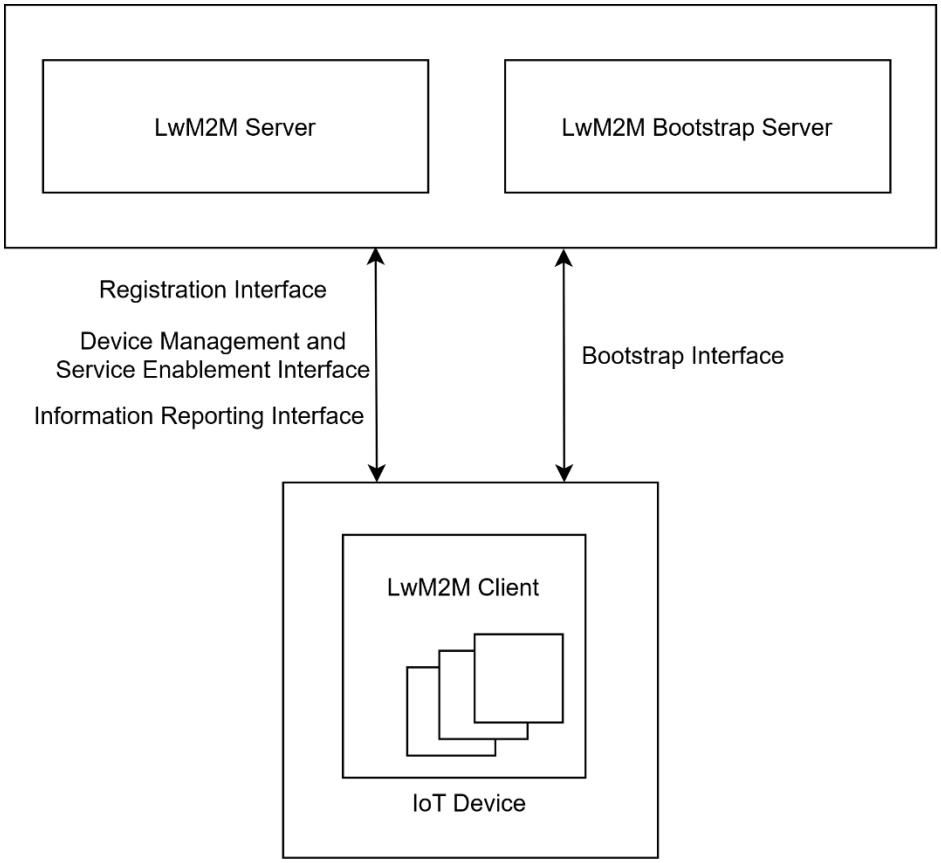

Figure 2.2: LwM2M Architecture [7, p. 20]

Figure 2.2 shows in which context the interfaces are located: Registration Interface, Device Management and Service Enablement Interface, Information Reporting Interface involve LwM2M Server and LwM2M client, while Bootstrap Interface regulates the LwM2M Bootstrap Server/LwM2M Client flow. The next sections will describe the four interfaces in detail.

2.1.3 Bootstrap Interface

The security credentials and certificates provisioning take place in the Bootstrap interface. The main actor here is the Bootstrap Server, which is an entity providing trusted certificated for the later Client-Server communication. It acts indeed as Key Distribution Center for the client, but it is not mandatory for every LwM2M scenario. Client and Bootstrap Server also communicate using the RESTful model. Some of the typical operations include: Bootstrap-Request, Bootstrap-Write, Bootstrap-Read, Bootstrap-Delete and Bootstrap-Finish. These are used to modify the Security Object within the LwM2M Client (for more information about objects, please refer to Chapter 2.1.7).

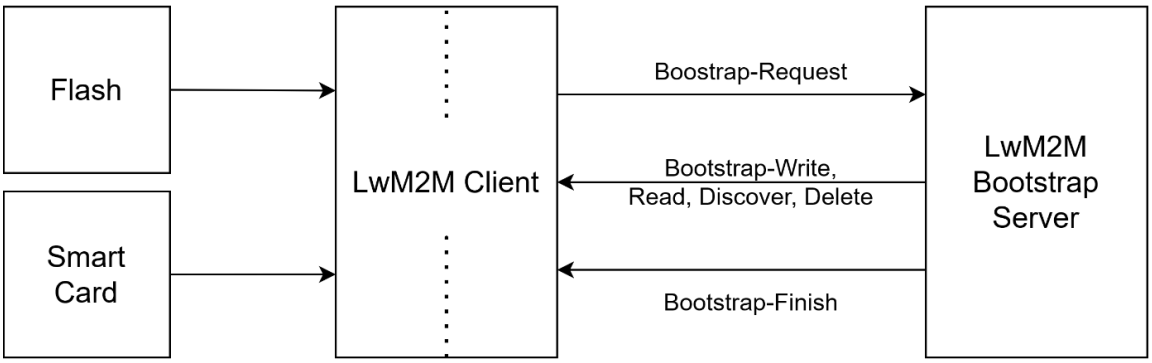

Figure 2.3: Bootstrap Interface Overview [7, p. 28]

There are four different modes within the Bootstrap Interface and two of them do not require the Bootstrap Server:

a. Factory Bootstrap

In this case the manufacturer provisions the IoT device a-priori with all of the information required to establish a secure Client-Server connection. The Bootstrap Server is not necessary in this context.

b. Bootstrap from Smartcard

This is a very similar approach compared to the Factory Bootstrap. The credentials-provision succeeds through a Smartcard to simulate a plug-and-play process. A secure channel between IoT device and card should exist, as mentioned in the LwM2M 1.2 Core Specification [7]. This approach does not include the Bootstrap Server neither.

c. Client Initiated Bootstrap

Figure 2.4 summarizes the transmission flow between Client and Bootstrap Server in this manner:

0. The LwM2M Bootstrap Server has some sort of Database knowing that a specific client should receive a predefined certificate or key. This configuration is out of the OMA specification’s scope
1. The LwM2M Client has manufacturer-provided credentials and knows which bootstrap server should be contacted
2. The LwM2M Client initiates the Bootstrap by sending a request under DTLS. This also contains the necessary certificate to authorize the Bootstrap Server
3. The LwM2M Bootstrap Server responds with objects containing the private key and the certificates necessary for the Client-Server Intercourse as well as the server’s endpoint to contact

Notice that the key distribution between LwM2M Bootstrap Server and Server is also out of the LwM2M specification’s scope. This may be implemented thanks to an API.

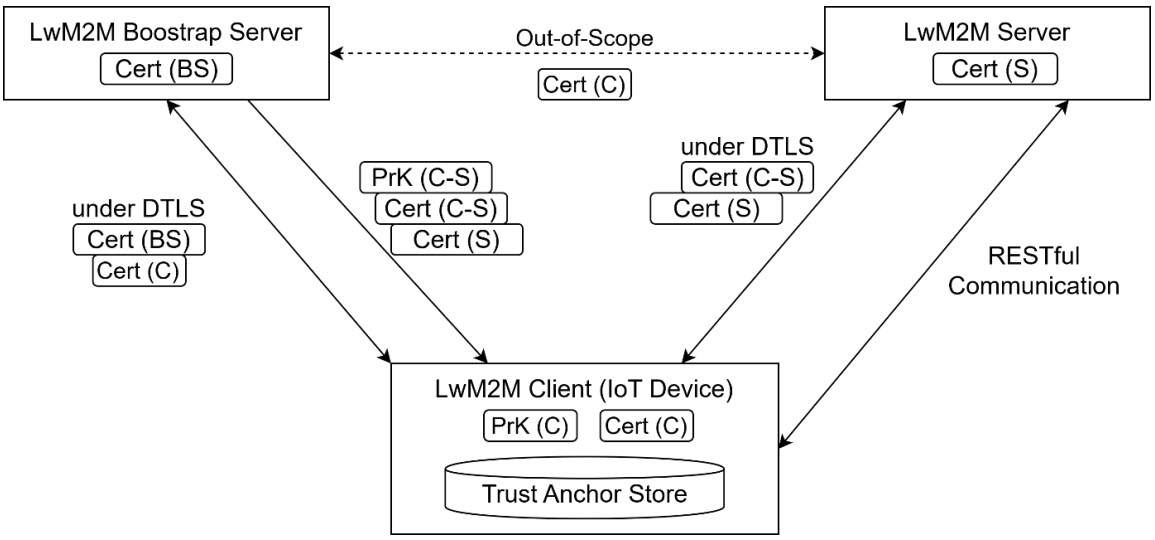

Figure 2.4: Client Initiated Bootstrap

d. Server-Initiated Bootstrap

This mode allows the start of the bootstrapping process thanks to a trigger mechanism initiated by the LwM2M Server as depicted in Figure 2.5. This is the only main difference to the client initiated Bootstrap mode and was created as an alternative to other protocols for configuring the Bootstrap credentials within the LwM2M Client. This method cannot work without a pre-existing connection between server and client. Notice that in previous LwM2M specifications, the LwM2M Server was meant to create and send to Client its Bootstrap raw and public keys; this was due to the insufficient Deterministic Random Number Generator (DRNG) capabilities of some constrained devices. Nevertheless, OMA decided to abandon this because of security and NAT related problems that did not allow the Server to always reach the client.

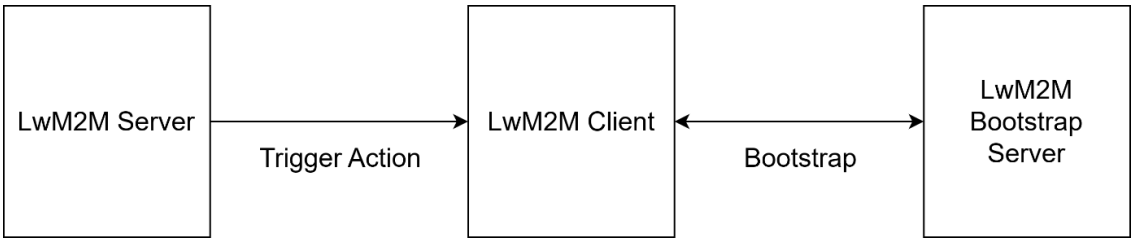

Figure 2.5: Server-Initiated Bootstrap

2.1.4 Registration Interface

Once the Bootstrap process is over, the registration interface regulates the handshake between LwM2M Client and Server as shown in Figure 2.6. The operations involved are: Register, Update and De-register. In the first one the Client provides its Objects and Resources as well as the its endpoint name. The second one is triggered when the lifetime expires or a certain event takes place, i.e. when a resource is modified. Finally, the De-registration ends the Client-Server connection.

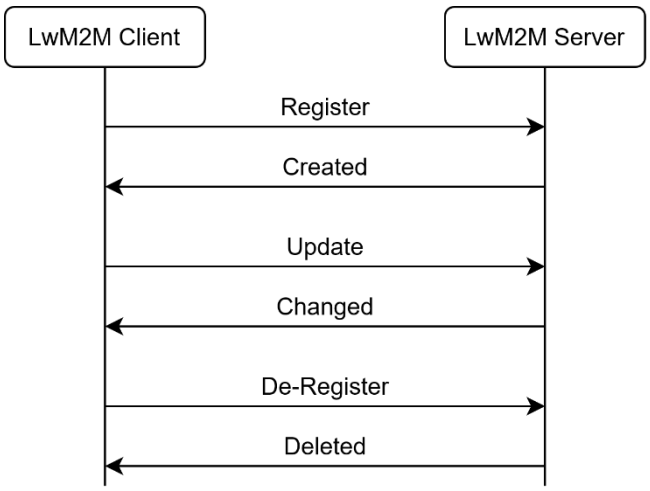

Figure 2.6: LwM2M Client-Server Registration Flow [7, p. 42]

2.1.5 Device Management and Service Enablement Interface

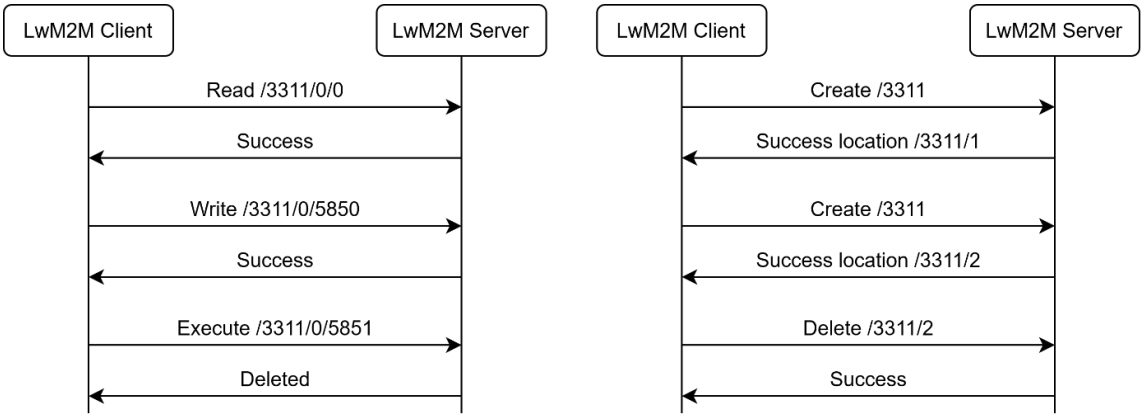

Figure 2.7: Client-Server request/response flows in the Device Management and Service Enablement Interface [7, p. 54]

Figure 2.7 represents request/response operations within the Device Management and Service Enablement Interface. On the left the server reads, writes, and executes resources while on the right it creates and deletes object's instances (for more information about objects and resources please refer to Chapter 2.1.7). This interface allows synchronous communications. Create, Read, Read-Composite, Write, Write-Composite, Delete, Execute and Write-Attributes operations may refer to one or multiple (Composite operations) resources/objects/instances [7, p. 54]. The only exception is the Discover request where the LwM2M Server demands for the Client's objects description. Another uncommon operation is the Write-Attributes. Here, the Server modifies the resource's attribute to state conditions for the asynchronous communication. The next section enlightens more about this topic.

2.1.6 Information Reporting Interface

The LwM2M standard introduces a new interface for dealing with asynchronous operations. The Information Reporting Interface is indeed responsible for deliver information similarly to the publish/subscribe concept in MQTT. As Figure 2.8 describes, the Server decides which object and/or instances it would like to observe while the client notifies back when a condition is met (here the Server decides to observe /3311/0/5850 which is the sensor's temperature resource). This is defined thanks to the write-attribute request as described at the end of Chapter 2.1.5 by declaring the fields Property and

Notification. Properties specify the dimension, the number of resource's instances, and the version, in case objects would have more instances. On the other hand, Notifications represent the notification's condition: minimum and maximum notification time-interval, greater than or less than a resource's numeric value, and step that states a value-difference below which the device will not send the new data. In this case, the previously set condition is the minimum and maximum time-period (in seconds) for triggering a new message, p_min= 600 and p_max = 780 seconds, between 10 and 13 minutes. In Figure 2.8 the first temperature variation happens before p_min causing the message to be queued until reaching 600 s. The second one appears between p_min and p_max and this allows an instant notification. Finally, since there is no new incoming temperature the Client notifies the Server after 780 seconds with the old value. The operations used in the Information Reporting interface are therefore: Observe, Observe-Composite, Cancel Observation, Cancel Observation-Composite, Notify and Send. The last one plays a role when the client notifies information without having received any specific observation request.

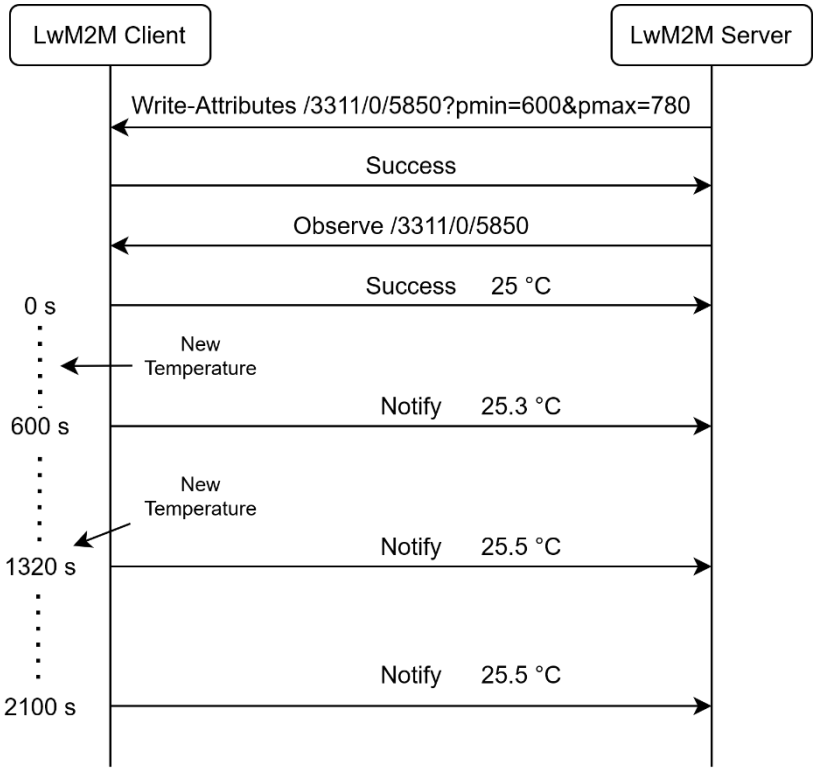

Figure 2.8: Temperature Observation by the Server in the Information Reporting Interface [7, p. 65]

2.1.7 Object Model

Every IoT device has one or more features integrated. As an example, a fire-alarm may have a flashing light, loudspeaker, smoke sensor, internal battery, and other components. Because LwM2M is a device management protocol, these properties are organized in a logical manner: the previously mentioned fire-alarm elements are called Resources and have an identifier. As Figure 2.9 shows, these Resources are then organized in Objects: flashing light, loudspeaker and smoke sensor belong to the fire-alarm object while internal battery, manufacturer, and serial number to the device object. Moreover, every Object and Resource may have multiple versions or copies of itself: these are called Instances. On Figure 2.9 you can notice instance number 0 and 1 of the fire-alarm object as well as the multiple ones belonging to the loudspeaker, smoke sensor and battery level resources. They are useful for extending the IoT device and in the same time keep the

old properties in case the Server or Bootstrap would not support the new ones. The URI format helps describing Objects, Resources, and Instances. For example, address 3/0/9/0 would denote the device object, its first instance, battery level resource (defined as 9 in OMA's device and resource registry) at its first instance. Every Resource may be then instantiated by the LwM2M Server using one of the following formats according to LwM2M version 1.2: plain text, opaque, TLV, JSON, CBOR, SenML JSON, and SenML CBOR.

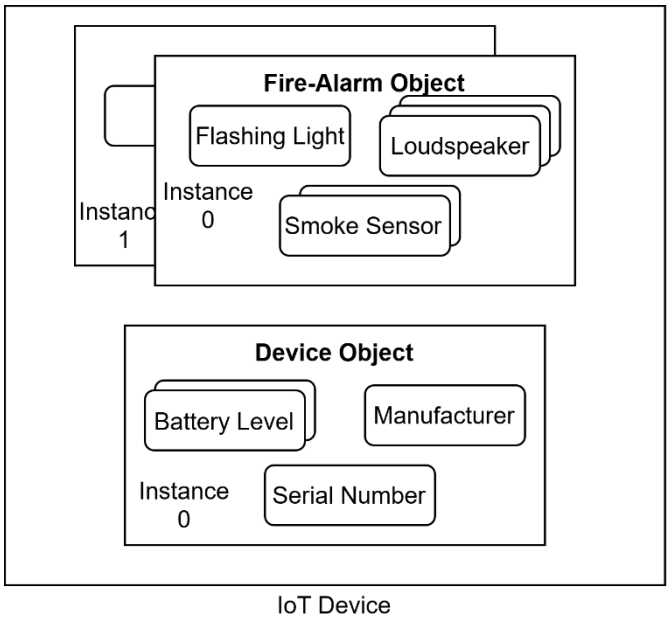

Figure 2.9: Example of Object Model in LwM2M

LwM2M specifies default objects that are mandatory in every Client. Table 2.1 reports them with the associated fixed Object ID. OMA has also decided to limit the number of Objects and Resources available within the specification. The LwM2M Registry contains many objects already available. Organizations, companies as well as private people are also allowed to reserve an Object ID range for their purposes. Nevertheless, OMA oversees accepting or refusing the creation of new Objects due to redundancy or inappropriateness.

Object Version	Object ID
LWM2M Security v1.2	0
LwM2M Server v1.2	1
LwM2M Access Control v1.1	2
Device v1.2	3
Connectivity Monitoring v1.3	4
Firmware Update v1.1	5
Location v1.0	6
Connectivity Statistics v1.0	7
LWM2M OSCORE v1.1	21
LwM2M COSE v1.0	23
MQTT Server v1.0	24
LwM2M Gateway v1.0	25
LwM2M Gateway Routing v1.0	26
5GNR Connectivity v1.0	27

Table 2.1: LwM2M Default Objects

Finally, the LwM2M Access Control List (ACL) provides an access control mechanism per instance. Every Object has its Access Control Object (ACO) whose ACL restricts the operations on Objects, Resources, or Instances for the LwM2M Server. Figure 2.10 represents the fire-alarm's ACO. The reported resources have different kind of permissions, but the ACL introduces Server permissions on top of them. In this situation, although Server 1 is allowed to read and write every resource, it will not be able to write on the loudspeaker resource since it only consents the read operation.

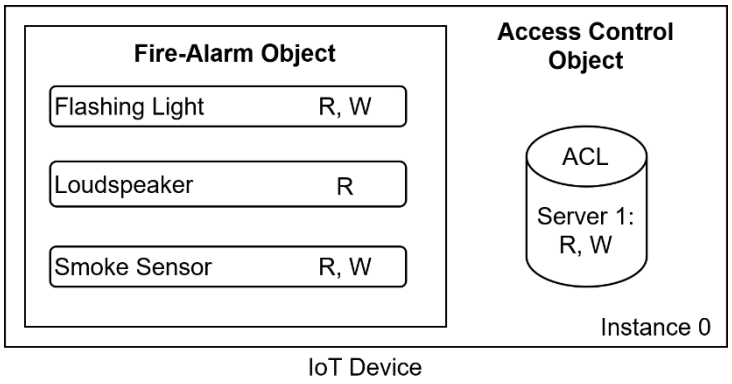

Figure 2.10: Access Control Object with its ACL [7, p. 69]

2.1.8 Security

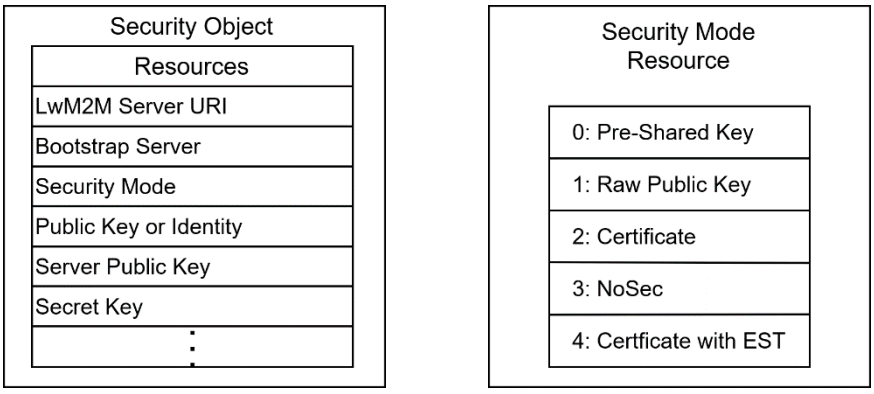

Figure 2.11: Security Object (left) and Security Mode Resource (right)

As mentioned in the protocol stack section, because LwM2M is also running on top of CoAP it also supports DTLS. The credentials used for authenticating the DTLS client and the DTLS server to secure the communication between the LwM2M Client and the LwM2M Server are obtained during the bootstrap phase. This step is particularly important since the Client must use different key pairs for different Servers [7]. When the Bootstrap Server distributes the keys to the LwM2M Clients, the Security Object represents a central role: it stores the a-priori credentials for the DTLS channel with the Bootstrap Server as well as the Server's keys for initiate the request/response phase. In case of a device update (also fundamental in IoT protocols) it protects the connection with a firmware-repository. Figure 2.11 left gives an overview of the security object. The Resource Security Mode (Figure 2.11 right) allows one between the following credentials mode:

0. **Pre-Shared Key**: this an asymmetric security mode that is more efficient and requires low computational effort during encryption and decryption
1. **Raw Public Key**: this is a way to use asymmetric keys avoiding the high complexity of certificates. "With raw public keys, only a subset of the information

found in typical certificates is utilized: namely, the SubjectPublicKeyInfo structure of a PKIX certificate that carries the parameters necessary to describe the public key" [8]

2. **Certificates**: a certificate Authority must control the authenticity of the certificates; this added to the creation and computation of PKs makes this asymmetric procedure the most complex and secure. This mode may create overhead in constrained IoT device. It is, therefore, not always recommended
3. **NoSec**: no-DTLS scenarios should be avoided in production. This option may be useful during testing, when applying security out of LwM2M, or in a controlled environment behind a gateway
4. **Certificates with Enrolment over Secure Transport (EST)**: this is the most secure and it is part of DTLS 1.2. EST is a service between Client and CA providing a secure channel to generate certificates. An important prerequisite is that the client must generate the private key locally. This underlines the need of a high DRNG. Choosing this method may cause overhead-over-the-air and should be compared with the real security benefits [7]

2.1.9 Sleep Mode

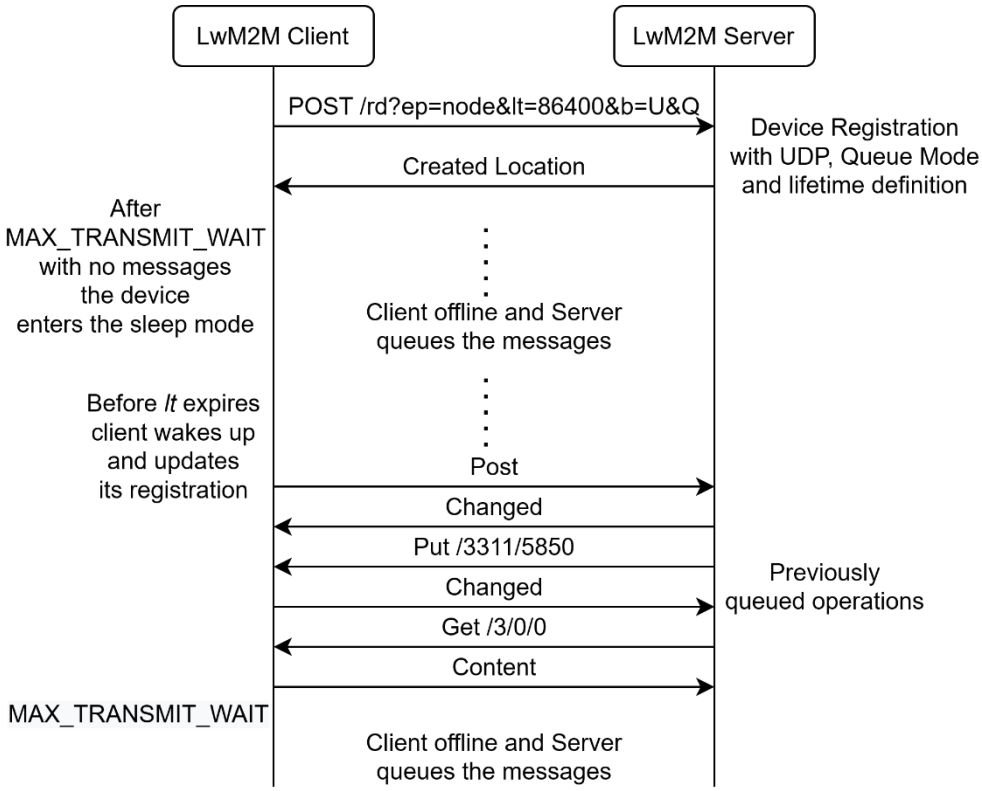

Figure 2.12: Sleep Mode Flow [9, p. 54]

Because of dealing with constrained IoT devices, LwM2M supports sleeping endpoints. In modern cellular IoT, remote devices are powered by an internal battery that must survive for several years. It is therefore fundamental to reduce the input power by entering the sleeping mode. As the communication flow in Figure 2.12 shows, the LwM2M Client communicates its sleeping cycles with the variable lt (lifetime). While the device is offline, the Server queues the messages for the whole lifetime duration. Once the Client wakes up and updates its status by sending a post operation, the Server is ready to send all of the previously buffered data. After exceeding the predeclared no-messaged time-frame after which the Client will enter the sleep mode, the Server restarts queuing its requests.

DTLS 1.2 CID and DTLS 1.3 comes with the connection ID (CID) feature. “Particularly when combining DTLS with the newly developed connection ID feature an existing DTLS session can be kept alive for a long time and the need for repeated handshakes can be avoided” [10]. This is extremely relevant for modern IoT constrained devices that want to establish a lightweight DTLS session. In fact, when a device usually wakes up, it has already lost its NAT binding. This is causing the impossibility by the server to decrypt the incoming messages: a new handshake causing unnecessary traffic would be needed. Supporting the newer versions of DTLS is therefore important to reduce traffic and battery consumptions.

2.2 MQTT

Message Queuing Telemetry Transport (MQTT) is a popular IoT messaging protocol. It is a Machine-to-Machine (M2M) connectivity standard that particularly suits to simple networks architectures [9]. In fact, with a minimum configuration it ensures stable transport of information between entities. These are represented by clients and brokers. The firsts are the endpoint where the information (also called message) can be computed and used for other scopes. The seconds are small servers which read messages incoming from the clients and routes them to the correct destination. This mechanism bases on the publish/subscribe principle where devices publish messages on specific topics. Referring to these, the broker then transports the messages to all the client which are subscribed to the same topics. MQTT is particularly meaningful when dealing a with a large number of devices reporting periodically an information such as the status of a light sensor. Those use-cases which involve constrained, non-performant and easy-to-deploy devices transporting lightweight information are the most appropriate in a network adopting MQTT. Simplicity is the feature that increased its popularity among individuals as well as established IoT device management platforms. Broker and Client open-source libraries like Mosquitto and Paho (with this last one also configurable at higher level with Python) made the standard accessible for users willing to enter the IoT world. The same is true for business cloud platforms supporting device management: AWS and Azure adopted MQTT as the first things connectivity protocol [11]. In 2019 the Organization for the Advancement of Structured Information Standards (OASIS) introduced version MQTT 5.0 with the following enhancements in respect to version 3.1.1 [12]:

- User properties: these are information about the device where to specify capabilities or requirements. They are integrated in the connection and/or in the publish messages and are UTF-8 encoded strings. The standard allows an unlimited number of user properties
- Payload format indicator: as the MQTT standard does not define any specific supported format, version 5.0 introduces the possibility of specifying if the payload is in UTF-8 format by setting a 1, or unknown by writing a 0 in the publish packet header
- Shared subscriptions: within scalable solutions it is important to establish a loud balancing behavior towards the clients. MQTT 5 introduces therefore shared subscriptions where just a client inside a group gets the message; subsequently for example the group’s back-end scales the message horizontally
- Reason codes: until MQTT 3.1.1 a broker could disconnect or reject clients without an explanation ACK. Now codes at application level indicates within the message’s header the cause of the interruption

- Session management: the broker has now the possibility of expiring a session with a client. By inserting a flag, the client indicates if the communication starts cleanly or by restoring a previous session

2.2.1 Protocol Stack

MQTT is built on top of TCP/IP "or over other network protocols that provide ordered, lossless, bidirectional connections" [9, p. 1]. The other suitable protocols are TLS (as a secure version of TCP) or WebSocket. A non-secure communication takes place in IANA's registered port 1883 while a secure session over port 8883. Figure 2.13 shows the MQTT protocol stack. It may build on top TCP or WebSocket. On the other hand, when secured it may rely on TLS. An MQTT's sister standard not anymore under maintenance is MQTT-SN. It operates on top of UDP and suits especially to those links where even lighter payloads are required and where the available bandwidth is limited. Wireless networks like BLE or IoT cellular standards like LoRaWAN or NB-IoT may benefit of a connections-less protocol like UDP. Nevertheless, MQTT-SN seems nowadays less relevant due to a difficult adaptability to various kind of wireless links as well as a non-maintained open-source project.

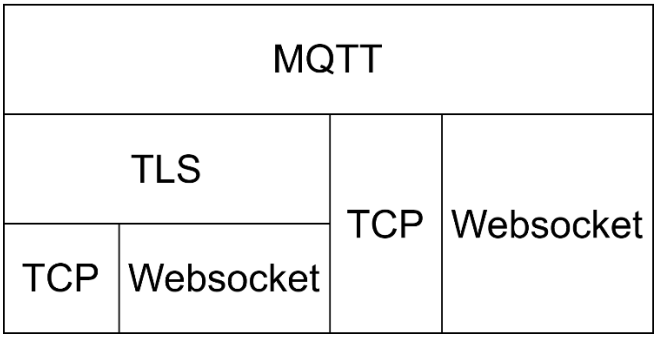

Figure 2.13: MQTT Protocol Stack

2.2.2 Network entities and Architecture

a) MQTT Client

Any endpoint represented by any kind of device may be defined as an MQTT client. It is indeed the receiver and/or sender of messages. The client is responsible for subscribing or publishing a topic. It also defines the payload format and the quality of service (Chapter 2.2.4) to send the message with. As specified in the introduction, MQTT deals optimally with constrained devices which do not offer high performances. This feature fits perfectly with the operation required by MQTT.

b) MQTT Broker

Acting as a distribution server, the MQTT broker is a central entity between the clients. It acts as a hub and its main task is to forward messages to the client. The condition of letting a packet through is to verify if the receiver is subscribed to the related topic.

c) MQTT's Architecture

Any MQTT topology includes at least two clients as well as one broker. Figure 2.14 takes in account a scenario with a broker in the middle and different subscribers and publisher depicted as devices or clients. The curtains' actuator which is interested the light status inside the kitchen subscribes to the topic kitchen/light. After the subscription the actuator will be notified every time the light sensor in the kitchen reaches or passes a certain value

in order to close the curtains. This happens because the kitchen's light sensor will constantly notify a new value through the kitchen/light topic. The same holds true for kitchen's main lamp which wants to know as soon as the room becomes dark and the curtains are not closed. Finally, also the light sensor receives the status of the lamp through the kitchen/lamp topic in order to know whether the brightness was caused by the lamp itself or not. Please notice that clients may be publishers and subscribers at the same time.

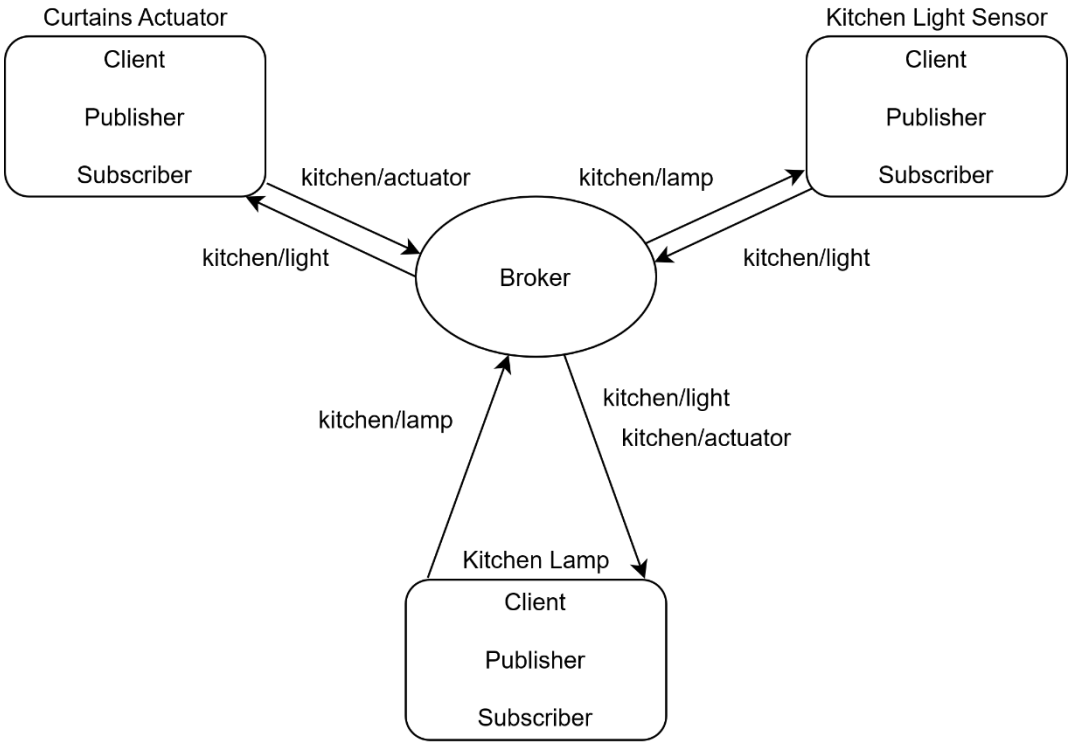

Figure 2.14: Example of an MQTT Architecture

2.2.3 MQTT Control Packet format

A packet exchanged under MQTT is divided into three parts: fixed header, variable header, and payload. The latter is not necessarily present in every MQTT packet [9, p. 21].

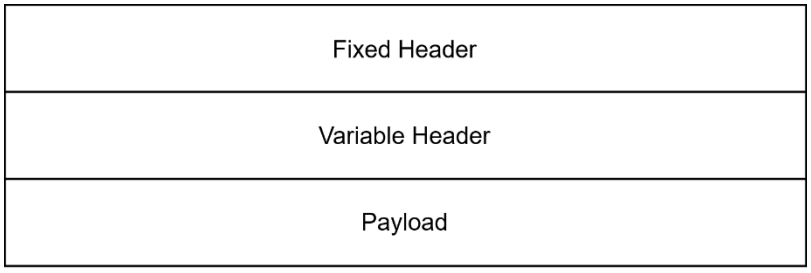

Figure 2.15: MQTT Control Packet Format [9, p. 21]

a) Fixed Header

As depicted in Figure 2.16, the fixed header consists of 2 bytes divided in this way: the first byte is divided equally between control packet type and flag specific to the control packet type; the second one corresponds to the remaining length. An overview about the components consist of [9, pp. 21-23]:

- Control packet type: this is the type of message sent. Some of the types included are: CONNECT, CONNACK, PUBLISH, PUBACK, SUBSCRIBE, SUBACK, DISCONNECT and AUTH
- Flag specific to the control packet type: as of MQTT 5 this is just relevant when associated to a PUBLISH packet type. It refers to the Duplicate delivery of a PUBLISH packet (DUP), PUBLISH Quality of Service (QoS) and PUBLISH retained message flag (RETAIN)
- Remaining length: it is equal to the remaining bytes between the fixed header, including the effectively taken bytes in the variable header for encoding, and the payload

Bit	7	6	5	4	3	2	1	0
Byte 1	Control Packet Type				Flag specific to the Control Packet Type			
Byte 2	Remaining Length							

Figure 2.16: Fixed Header [9, p. 21]

b) Variable Header

The MQTT variable header consist of a two-byte integer packet identifier plus two byte variable length divided between property length and property [9, pp. 23-26]:

- Property Length: it specifies the length of the property field by defining a variable byte integer
- Property: it is also represented by a variable byte integer and for each hexadecimal value it identifies a usage, data type and the packet for which it may be required. An example of property is the reason code in the newly introduced MQTT 5. Table 2.2 reports some of the properties.

Hex	Usage	Type	Packet
0x0B	Subscription Identifier	Variable Byte Integer	PUBLISH, SUBSCRIBE
0x1F	Reason String	UTF-8 Encoded String	CONNACK, PUBACK, PUBREC, PUBREL, PUBCOMP, SUBACK, UNSUBACK, DISCONNECT, AUTH
0x27	Maximum Packet Size	Four Byte Integer	CONNECT, CONNACK

Table 2.2: Example of Properties [9, pp. 25-26]

c) Payload

Finally, the payload contains the data any subscriber is interested in. Just some packet types require it: CONNECT, SUBSCRIBE, SUBACK, UNSUBSCRIBE and UNSUBACK [9, pp. 26-27].

2.2.4 MQTT Quality of Service (QoS)

Since MQTT may rely on TCP/IP, a connection-oriented transport protocol, it also can count on a certain level of consistency. In fact, TCP's three-way-handshake ensures that the message arrives at the destination by delivering an acknowledgement to the sender. This is true for a topology involving to two entities, client, and server. MQTT on the other hand has a minimum network composition of two clients plus a broker/server. In this case the number of the messages needed for a PUBLISH increase proportionally with the number of subscribers involved. Especially for a protocol designed for constrained devices with a small battery like MQTT, it is fundamental to keep the number of packets as small as possible. In order to achieve this, the standard introduces different QoS levels which denote how many times a subscriber receives a particular message. The client is responsible for defining the QoS for a specific subscription or publication. This means that a hypothetical communication involving one publisher and one subscriber, if the first sets an higher QoS than the second, the broker will downgrade the QoS level choosing the lower one [9, p. 76]. Notice that by controlling the QoS within MQTT there is no change in how the underlying protocol works including error detections or error corrections [13, p. 13].

a) At most once – QoS 0

In very stable network condition or when the user wants to preserver the devices' battery-life, the MQTT messages may be transmitted with QoS 0. At least once means that the payload is sent to the subscriber only and at most once, it is not stored, and no confirmation is sent to the publisher. The terms "fire and forget" is often used to define this behavior since the MQTT network may just lose track of the message [14].

b) At least once – QoS 1

This is the default QoS in MQTT. It consists in delivering the message to the subscriber at least once. If the sender does not receive any confirmation of reception by the receiver it will generate a duplicate message adding the DUP flag in the packet. While waiting for the acknowledgement other devices may take initiative and publish payloads the previous mentioned receiver is subscribed to. This would cause multiple incoming messages by the subscriber and acknowledgements would eventually get lost. Therefore, it is possible that duplicate messages may arrive. This is why QoS 1 is referred as at least once [14]. This level is for scenarios where device can deal with duplicates or where there is a need of balance between speed and delivery reliability.

c) Exactly once – QoS 2

When the overhead is not relevant and it is fundamental that clients do not receive duplicates, QoS 2 ensures an accurate transmission. In fact, like Figure 2.17 describes there are two rounds of transmission. The first one happens in a similar way as in QoS 1, where the publisher waits for acknowledgement. In this case however, the publisher stores the message and waits before taking further actions. Only after the acknowledgment (PUBREC) the second round takes place: the publisher sends a confirmation (PUBREL) for allowing the subscriber to process the data. Then the sender waits for the subscriber's acknowledgement to this second round (PUBCOMP) and only after receiving it, it deletes the stored message. Notice that the receiver keeps a reference of the packet identifier to avoid processing the data for the second time in case another PUBCOMP would arrive. [14, 15].

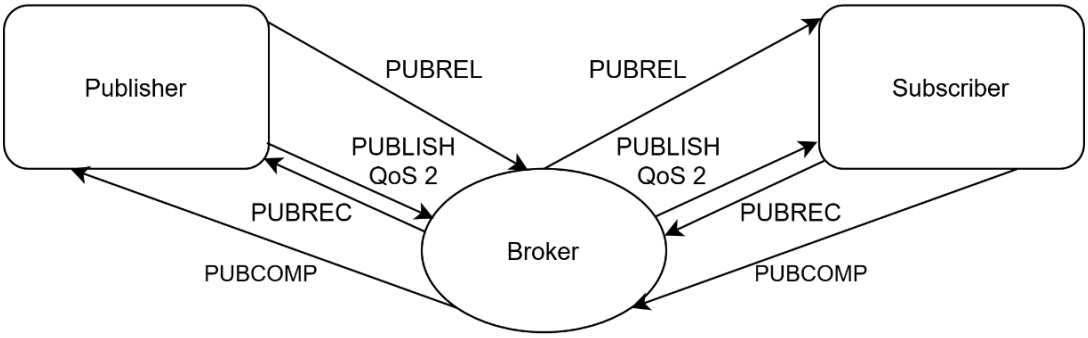

Figure 2.17: QoS 2 workflow

2.2.5 Security

"MQTT is a transport protocol specification for message transmission, allowing implementers a choice of network, privacy, authentication and authorization technologies. Since the exact security technologies chosen will be context specific, it is the implementer's responsibility to include the appropriate features as part of their design" [9, p. 111]. This is the first statement of MQTT 5 security's Chapter in the specification and gives a clear understanding about MQTT's strategy: the implementer is the one who is responsible for enhancing its solution's security. This allows a certain freedom in choosing a use-case specific environment but leaves at the same time any default implementation without neither authentication nor authorization mechanisms. A non-secured MQTT's configuration has the following vulnerabilities [16]:

- Lack of robust authentication: MQTT's standard includes the possibility of using a username and password for authentication. Until version 3.1.1 these were part of the CONNECT message in plain text allowing the client to authenticate the broker and vice versa. MQTT 5 newly introduced the AUTH message (also configurable): client and broker first verify the dually supported authentication method by inserting the property field in the CONNECT and CONNACK messages. After, by exchanging AUTH packet types they authenticate each other with the credentials which are unlike in version 3.1.1 sent in form of tokens and not anymore in plain text. The choice of the authentication method is not limited: MQTT's standard mentions SCRAM and Kerberos but there is no restriction [9, pp. 106-108]. The AUTH message is enhancing the solution's security but still does not force the developer to include authentication methods within a secure communication protocol like TLS and SSL. Also, the support of those methods is highly dependent on the device's capabilities

- Lack of robust authorization: after authenticating the broker, clients are by default allowed to subscribe or publish on any topic they want without any restrictions. The broker is itself responsible for consenting clients to use certain topics. One solution may be topic permissions on broker side where restrictions may occur; but again, this is not included in a non-configured solution

- Lack of confidentiality: MQTT operates by default thanks to unencrypted transport protocols like TCP. This leaves any communication easily readable for any attacker. Any tool for packet tracing would be sufficient to read the payloads between client and broker. The most immediate solution would be using TLS

- Lack of integrity: as for lack of confidentiality the problem here lies in the unsecured transport protocol. Any TCP communication may be in fact easily modified

by unauthorized clients using unauthorized topics. A resolving method would be the integration of Checksums, MAC, or digital signatures in the protocol

A possible configuration of security mechanisms within MQTT at different OSI layers would include:

- Network layer: VPN
- Transport layer: SSL/TLS using PSK or certificates
- Application level: username/password credentials with AUTH method in MQTT 5 and broker topics authorization

2.3 LTE-M

LTE for Machine-Type Communication (LTE-M) is a variant of LTE with the following attributes:

- Modem and Equipment Cost reduction
- Coverage enhancement
- Improved devices' lifetime
- Scaling LTE devices

2.3.1 Deployment flexibility

3GPP's releases 12, 13 and 14 gradually introduced Long Term Evolution features supporting Machine-Type Communication (MTC) and IoT. Release 12 introduced Cat-0, an initial study on how to enhance coverage for MTC User Equipment (UE). Release 13 focused on the physical layer and presented Coverage Enhancement (CE) A and B modes; devices following this specification are called Cat-M1. Finally, Release 14 and more recent ones improved performances specifying Cat-M2 devices. The purposes of these new releases were to decrease LTE costs for MTC (LTE-M). Cat-0 modem costs had to be reduced by 1/3 by reaching the following compromises: reduced peak rates, single receive antenna, half-duplex operation, reduced bandwidth (BW), and reduced maximum transmit power. For example, Release 12 goals was to reduce UL and DL data rate from 5Mbps and 10Mbps (Cat-0 UL and DL) to 1Mbps. On the other hand, Cat-M1 operates with reduced BW (from 20 MHz in Cat-0 to 1.4 MHz) and lower power class (from 23 dBm to 20 dBm). Another important goal was to achieve CE. Less stringent data rates and latency requirements allow to increase the coverage thanks to retransmission and repetition techniques. CE Mode A and B should help reaching a 20 dB CE. Along the already mentioned aspects, 3GPP achieved an improved devices' lifetime by introducing Power Saving Modes (PSM) techniques as well as with the enhanced Extended Discontinuous Reception (eDRX). Moreover, Radio Resource Control (RRC) Suspend/Resume techniques helps reducing signaling after reconnection between UE and BSS allowing an easier machine scalability. Finally, LTE-M deployment is easy due to the adaptability to the already existing LTE network. Shared resources as well as LTE less busy traffic periods are vital for the Cat-M expansion. [17, pp. 137-139]

2.3.2 Design Structure

a) LTE E-UTRAN and EPC Architecture

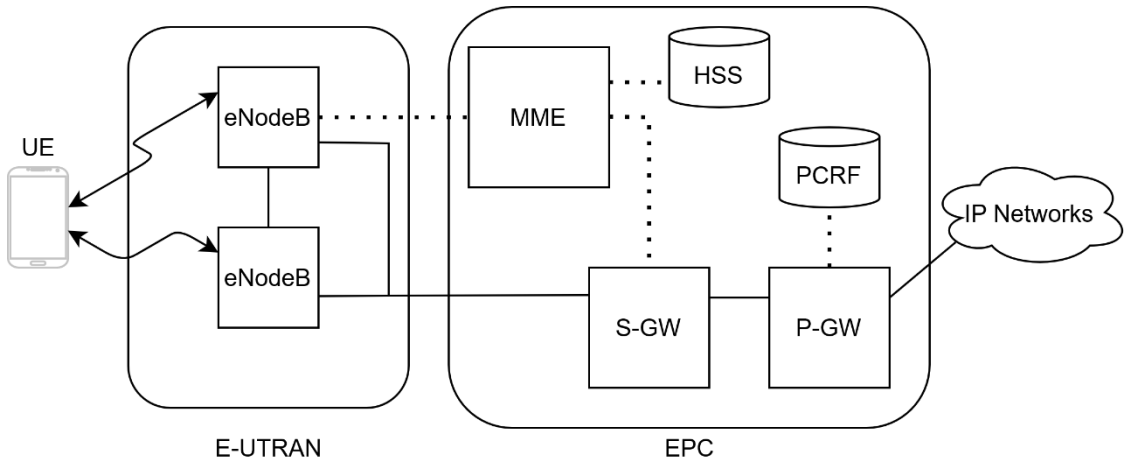

Figure 2.18: LTE Architecture

As LTE remains the base for LTE-M (and later on for NB-IoT) Figure 2.18 gives an overview of a simplified LTE network Architecture. One the left the UE or device communicates wirelessly with the eNodeB, contained in the base station (BS), which is a physical entity giving access to the LTE network. eNodeBs not only corresponds with the device but also with each other. The Evolved Universal Terrestrial Radio Access Network (E-UTRAN) represents the cellular physical access and interfaces. The successive high-level component is called Enhanced Packet Core (EPC). This is the core network of the architecture and contains the Mobility Management Entity (MME), Serving Gateway (S-GW), Packet Data Network Gateway (P-GW), Home Subscriber Server (HSS) as well as the Policy and Charging Rules Function (PCRF). The MME is the control interface between EPC and eNodeB. It is responsible for the signaling of the UE or eNodeB towards the EPC and it also allocates the gateways for accessing the internet. MME has also a security role: it generates temporary identifications for the UEs, authorizes the device to use the service provider's network, controls the roaming authorization and manages the key distribution center. Next to the MME, the HSS is a database which contains subscription-related data: every SIM or eSIM card is authenticated and authorized by consulting the HSS. The S-GW make sure to maintain the network session during handovers between eNodeBs as well as between LTE and other kind of cellular networks. It also generates paging requests towards the UE (for more information about paging please visit Chapter 2.3.8). Nevertheless, the P-GW represents the real point of entry for traffic incoming from and outgoing other packet data networks. A UE may be connected to more P-GW as the device creates more sessions with more different networks. Finally, the PCRF is responsible for service policy and Quality of Service (QoS). After SIP session is established it allows or rejects the media request, controls the allocation policies for a certain resource and manages the Packet Data Protocol (PDP) context for a new request. The traffic is in the end directed from the EPC to the external IP Networks [18].

b) Frames and PRBs Architectures

LTE-M keeps the same physical layer design as LTE. OFDM in DL and SC-FDMA in UL are still present. The modulation schemes are QPSK and 16QAM. Also, the frame and Physical Resource Block (PRB) architecture is the same as LTE. Referring to Figure 2.19 on top there are 1024 hyperframes, each of those is divided in other 1024 frames which contain 10 subframes each. These are then divided into two slots; each of them has a

duration of 0.5 seconds and represents from 6 to 7 OFMD symbols depending on the Cyclic Prefix (CP): normal CP supports a 4.7 µs propagation delay while extended CP up to 16.7 µs (rarely used).

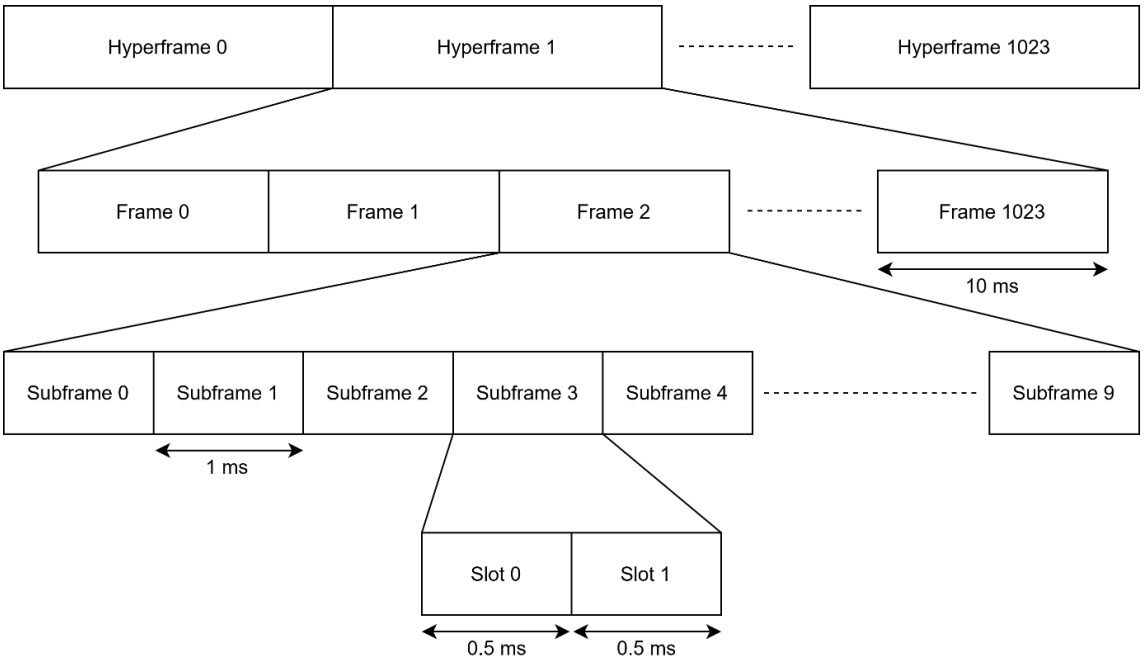

Figure 2.19: LTE and LTE-M frame architecture [17, p. 140]

As described in Figure 2.20 one LTE-M's PRB contains 12 subcarriers whose subcarrier spacing is 15 kHz representing a total of 180 kHz. One subframe slot has then 6 to 7 PRBs (in this case 6). [17, pp. 140-142]

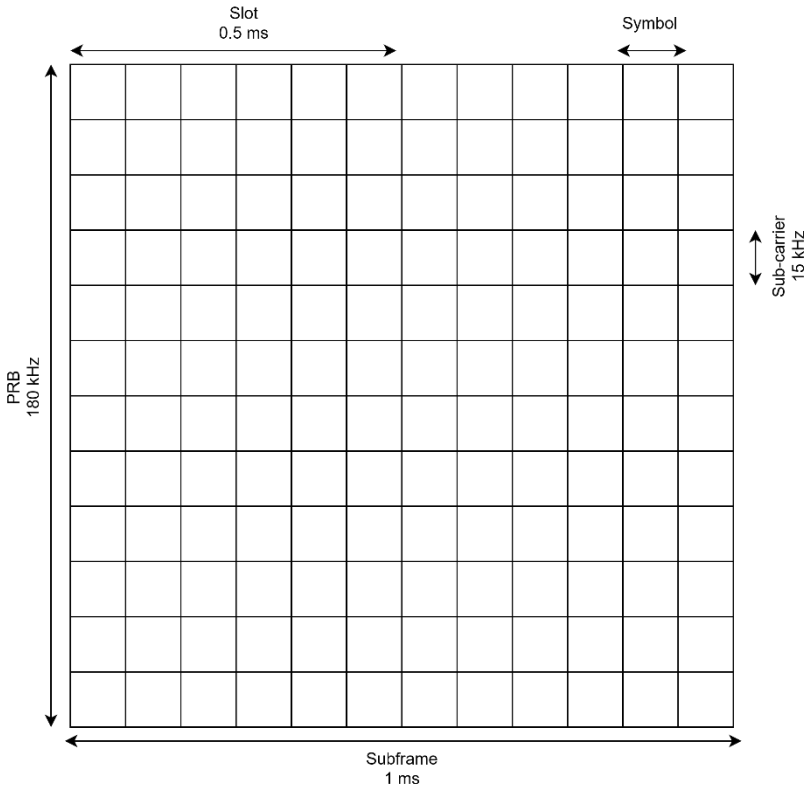

Figure 2.20: LTE and LTE-M PRB and subframe relationship [17, p. 141]

2.3.3 Duplex Mode

Frequency Division Duplex (FDD) and Time Division Duplex (TDD) are part of LTE-M. Here we can find the distinction between Full-Duplex (FD) where UL and DL happen at the same time, and Half-Duplex (HD) where UL and DL are separated. The latter may functionate using a double oscillator for carrier frequency generation (Type A) or a single one (Type B). LTE-M and Cat-0 devices use Type B HDD in line with the objective of lowering MTC costs. To resolve issues caused by a slow reference-frequency switching in the oscillator, LTE-M ensures for Type B devices a guard interval between UL and DL and vice versa. Regarding FDD, UL and DL take place in the same carrier frequency but in different time-periods. A special subframe (similar to guard interval) helps switching between DL and UL. The other way around is not necessary: the eNB is the only entity transmitting in DL and can therefore plan the users DL periods; on the other hand, during UL in order to synchronize the users it requires extra-time given by the special subframe. Nevertheless, as a matter of a cost/performance trade-off, the manufacturer may choose between FD or HD (Type B) -FDD or TDD. [17, p. 142]

2.3.4 Narrowbands

LTE operates in the system bandwidths 1.4, 3, 5, 10, 15 and 20 MHz. It holds a maximum of 100 PRBs for 18 MHz. LTE-M Bandwidth-reduced Low-complexity (BL) devices on the other hand, only support 6 PRBs channels. This means that the LTE bands are by occurrence divided into multiple narrowbands. This requirements show the interoperability between the two standards: LTE-M Cat M1 uses indeed 1.4 MHz narrowbands (Cat M2 and non-BL 5 MHz) within higher LTE BWs. Table 2.3 shows for example that the LTE 10 MHz BWs has 8 narrowbands, the 15 MHz 12 and so on. Since in almost all of LTE regular BWs there is not an even number of narrowbands, some PRBs are not used for LTE-M data transmission: these are reported in the last column of Table 2.3. They may be utilized "for LTE-M related transmissions on other physical channels/signals and for any ordinary LTE transmissions in the cell" as well as for NB-IoT anchor and non-anchors carriers as also reported in Chapter 2.4.2 [17, p. 144].

LTE Bandwidths	Number of PRBs	Number of Narrowbands	PRB not party of Narrowbands
1.4	6	1	None
3	15	2	3 = edges + center
5	25	4	1 = center
10	50	8	2 = edges
15	75	12	3 = edges + center
20	100	16	4 = edges x 2

Table 2.3: LTE-M Narrowbands and PRBs not belonging to any Narrowbands [17, p. 143]

2.3.5 CE Modes

LTE-M coverage enhancements are a direct consequence of the reduced requirements in terms of maximum signal power and number of antennas for Cat-M devices. The uses-cases are also often related to rural landscapes as well as basements and other locations where the coverage may not be optimal. For all these reasons 3GPP decided first to introduce CE mode A to close the gap with LTE devices performances and subsequently mode B to extend the use-cases. Mode A supports up to 32 subframe repetitions of the

data channel and is obligatory on Cat-M1 and -M2 devices. Mode B goes up to 2048 repetitions and is not mandatory on all devices; this mode allows reaching a coverage target of 20 dB. Cat-0 and -1 modems may optionally have one or more modes.

2.3.6 Idle Mode

When the device is neither transmitting nor receiving data it finds himself in one idle mode. This state indicates that there is no DL or UL operation: the modem is either busy in finding and/or synchronizing with a cell or in a complete inactive phase. The first case is about the cell selection and the consequent system information acquisition. The second one includes power saving techniques like eDRX and PSM.

2.3.7 Cell Selection, Reselection and System Information Acquisition

When an LTE Cat-M modem is powered up it starts searching for an LTE cell nearby to synchronize time and frequency with the carrier signal. Here the Primary Synchronization Signal (PSS) is transmitted every 5 ms in the 62 central subcarriers and returns the carrier frequency error (CFO). Since LTE-M deals with low-cost IoT devices with less effective oscillators, the CFO has large values for the initial cell selection: the imprecision "may be as high as 20 ppm (parts per million), corresponding to, for example, 18 kHz initial CFO for a 900-MHz band" [17, p. 167]. Therefore, LTE-M devices have a relatively high power-consuming performance during the initial cell selection rather in the reselection or non-initial selection phases. In a successive moment, the Secondary Synchronization Signal helps identifying the Cyclic Prefix (CP) that may be normal or extended and the duplex mode, either FDD or TDD. Along with this, the UE collects the DL system BW as well as the information scheduler for the System Information Block (SIB). LTE-M uses a special kind of SIB called Bandwidth Reduced (BR) that contains among others information about other frequencies and interfrequency neighboring cells relevant for LTE, UMTS 3G, GSM and CDMA2000 3G cell reselection [17, p. 170]. The Base Station (BS) is also acquiring data about the UE: country, cell reservation information, minimum required Reference Signal Received Power (RSRP) and Reference Signal Received Quality (RSRQ). These information also ensure LTE-M devices to support mobility: during cell reselection UEs can select a new BS based on the RSRP calculation.

2.3.8 eDRX

During the idle state, the network periodically monitors the UE’s state by an action called paging. The BS transmits paging records to all the devices connected to the narrowband. Every endpoint then controls if its Temporary Mobile Subscriber Identity (TMSI) or the International Mobile Subscriber Identity (IMSI) corresponds to the one contained in the paging message. If yes, the device initiates a connection with the cell. As a result, this procedure impacts on the battery-life duration as well as on the DL data rate. The solution is the extended discontinuous reception (eDRX), a cyclic event period dedicated to paging. The minimum eDRX period in LTE-M in both idle and connected states is 2.56 s while the maximum is 44 min in idle and 10.24 s in connected mode (when data transmission takes place). After this cycle occurs the UE opens a time-frame (maximum 20.48 s) in which it accepts paging records. For NB-IoT there are slight changes: for DRX (a previous version of eDRX) the longest cycle corresponds 10.24 s while in eDRX the maximum is 2h, 54 min and 46s (one hyperframe). Figure 2.21 represents on the lower part the eDRX cycle and above the paging windows which appear as soon as the device wakes up and enters the idle mode. The Mobile Management Entity (MME) is responsible for choosing the paging strategy: if based on the BS paging history the UE is not easily

reachable (i.e. located in a basement) the MME may decide to include repetitions for the transmission. This, on the other hand, is difficult for moving devices since the MME cannot rely on a cell-related paging history. In this case the device becomes responsible to generate MO traffic to support the MME.

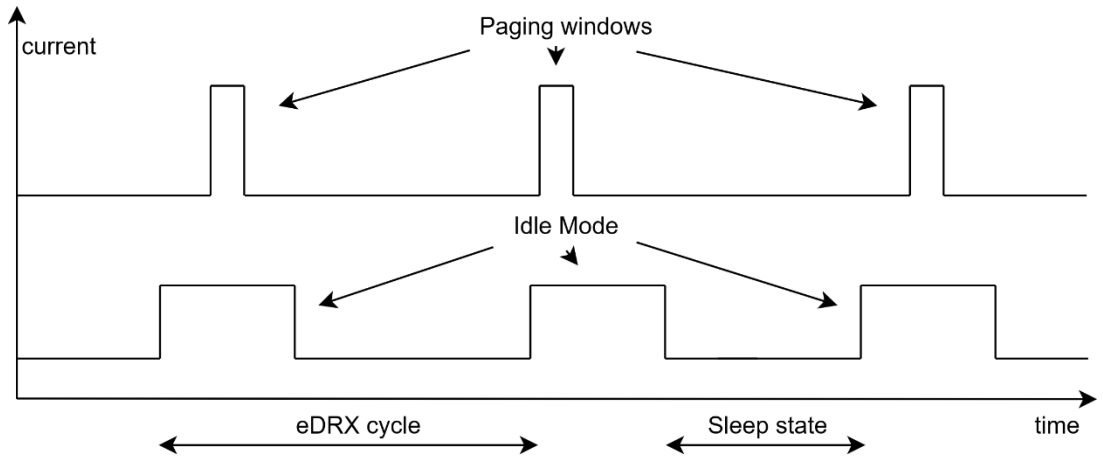

Figure 2.21: eDRX cycle and paging

2.3.9 DRX and RRC Inactivity Timer

When the UE is in connected mode and a data exchange occurs the successive transition into idle mode does not take place immediately. There is a time-frame when the eNodeB does not know if the UE will continue transmit data or stop for a longer period. Therefore, as illustrated in Figure 2.22 the RRC inactivity timer takes place. This is divided into DRX inactivity timer with a persistent active state followed by short and long DRX timer cycles. These are different than the regular DRX cycles (which only exist in idle mode) since the paging windows are now substituted by active states where the UE could instantly handle upload and download. Every provider in the world decides independently the RRC inactivity timer duration and composition. Some of them for example completely eliminate the DRX timers or only keep the long DRX timer. Also, the DRX inactivity timer duration variates between carriers. Nevertheless, the choice of these parameter may influence the devices' battery-life. For this purpose, modems' manufacturers started to introduce the so-called Release Assistant Indication (RAI) where the UE may signalize when the data transmission is over. This option is however still very limited within both modems and providers network. Moreover, it is mostly supported only within NB-IoT, at least in the EU.

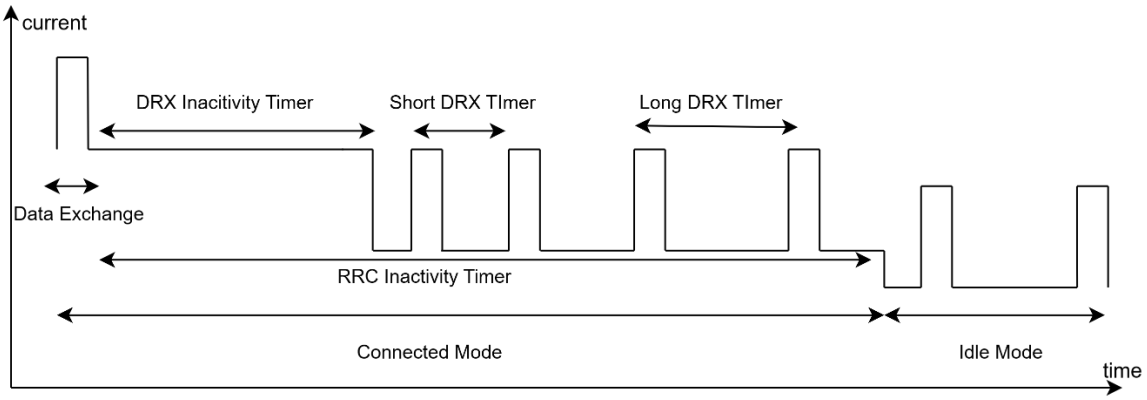

Figure 2.22: DRX and RRC Inactivity Timer

2.3.10 Power Saving Mode

Between the end-paging windows and the next eDRX cycle the user may decide to activate a time period where the device sleeps. This is part of the Power Saving Mode (PSM) settings, a feature present in all 3GPP IoT cellular standards. PSM is suitable for use-cases where a UE does not require data DL or UL for long time-periods (hours, days, months) and can therefore enhance its battery performances. Both GSM/EDGE and LTE incorporate PSM which devices use to drastically minimize their power consumption. Figure 2.23 depicts the PSM as the period where no communication takes place; at the bottom you can notice its time-frame taking place between the end of the idle mode and the beginning of the connected mode. PSM differs indeed from the idle state where paging is still considered: here any MT traffic remains ignored since no UE's component, but an internal timer, is active. The integrated clock helps to enter and leave the PSM mode. MO communications are the only exception for the device to leave PSM. There are three scenarios for which the EU may perform MO UL: data-transfer, Tracking Area Update (TAU) and Routing Area Update (RAU). These last two take place periodically and are fundamental to allow a temporary active state when the device accepts MT traffic. In LTE-M and NB-IoT TAU and RAU cycles may be between seconds and an entire year [17, p. 21]. After TAU and/or RAU, the eDRX (described in Chapter 2.3.8) and paging periods take place after which the active internal timer trigger the sleeping mode again. In PSM the UE remains registered to the BS allowing a lightweight wake-up signaling. The PSM benefits may immediately disappear if the device moves frequently and needs a new cell selection with consequent registration and attachment. These operations require much more power than RAU/TAU and have to be limited as much as possible to ensure UE's longevity.

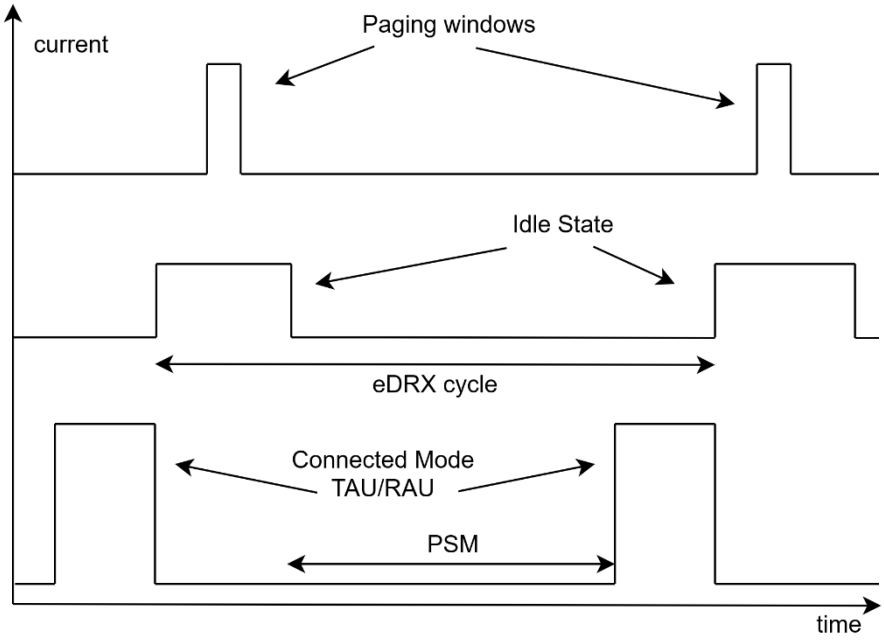

Figure 2.23: PSM cycle in combination with eDRX

2.4 NB-IoT

3GPP's release 13 was characterized by a study called "Cellular System Support for Ultra-low Complexity and Low Throughput Internet of Things" [19] which theorized a new IoT cellular standard with objectives similar to LTE-M: improved coverage, capacity, battery lifetime together will less stringent latency time than Cat M devices and lower device costs. Another important goal was to reutilize GSM and LTE spectra in order to avoid new networks and infrastructure which are normally expensive and complex to build. Narrowband Internet of Things (NB-IoT) is therefore the outcome of this. Tackling the devices' complexity and cost, 3GPP decided to relax the base-band operations by searching for only one synchronization sequence using a lower sampling rate (e.g. 240 kHz) when coupling time and frequency with the network [17, p. 220]. Moreover, in the channel coding NB-IoT abandons LTE's turbo codes and instead opts for simple convolutional codes like tail-biting (TBCC) in DL. Also, modulations schemes are only of low-order and not multilayer MIMO anymore, just supporting half-duplex to separate DL and UL operations to reduce hardware's complexity. Avoiding MIMO also means one single antenna required allowing a higher oscillator inaccuracy reaching 20 ppm (like in LTE-M). As in LTE-M, devices' transmit power level is limited to either 20 or 23 dBm, on-chip Power Amplifiers (PA) become beneficial [17, p. 221]. In terms of Coverage Enhancement (CE) since NB-IoT relies on QPSK and BPSK in the UL channel its waveform maintains a close to constant envelope in UL. This is particularly important because the PA may work at the maximum level without the need of a power back-off and the coverage takes advantage out of this. This aspect is then sustained by repetitions and smaller data rates. Also, the battery lifetime benefits in terms of efficiency out of the minimized PA's power back-off. PSM, eDRX and connected mode DRX (cDRX) are different options to keep the power consumption low. The next section gives a theoretical background regarding why bandwidths play a small role in bad coverage conditions.

2.4.1 Small Bandwidths and bad coverage

Shannon's capacity theorem states a clear relationship between power, noise, bandwidth, and capacity:

$$C = W \log_2\left(1 + \frac{S}{N}\right) = W \log_2\left(1 + \frac{S}{N_0 W}\right) \tag{1.2.4.1}$$

where C is the channel capacity, S the signal power, N the noise power, W the noise bandwidth and N_0 the one-sided noise spectral density. Notice that the noise bandwidth is equal to the signal bandwidth if Nyquist's 1st criterion is applied. Moreover, for bad coverage $\frac{S}{N} \ll 1$ and using the approximation $\ln(1 + x) \approx x$ for $x \ll 1$:

$$C = \frac{S}{N_0} \log_2(e) \tag{2.2.4.1}$$

This shows that the capacity (or also the data rate) only depends on the signal power (the noise power is usually not adjustable). The bandwidth vanishes and does not play any role with bad signal coverage. This is particularly significant and indicates that it is possible and advisable to allocate small bandwidth when a device utilized the UL channel. As it will follow NB-IoT uses 180 kHz bandwidth for good coverage but the situation changes when the signal conditions change [17, pp. 221-222].

2.4.2 Operations Mode

NB-IoT comes with three different deployments methods: stand-alone, in-band and guard-band. This aspect is relevant to save costs and complexity from creating a new infrastructure. The goal here is to use either refarmed GSM or LTE bands.

a) Stand-alone

It is possible to deploy a stand-alone NB-IoT band by refarming for example part of the GSM spectrum. According to 3GPP's requirements [20], a guard-band between the GSM and the NB-IoT carrier must exist. This band is recommended to be 100 kHz between GSM and NB-IoT spectra belonging to the same operator while 200 kHz if two providers are involved. This means that not only the operator needs to refarm one GSM carrier for NB-IoT's deployment, but it would require an additional one and half for the guard-band. The upper part of Figure 2.24 takes in consideration a 200 kHz GSM carriers whose operator decides to introduce cIoT bands. The new NB-IoT band must be integrated in the entire 200 kHz band even though it is just 180 kHz wide. Therefore, beneath operator number 2 inserts a 100 kHz guard-band next to the regular GSM carrier as well as 200 kHz between operator number 1 and itself. Refarming GSM with the stand-alone deployment may result in a temporary loss of spectra and overall band inefficiency. Nevertheless, on a long-term GSM to LTE (or to 5G) migration plan it turns out to be particularly convenient. The momentary stand-alone deployments may become in-band or guard-band, allowing a step by step GSM refarming (please for more information refer to the next two subchapters) [17, pp. 222-223].

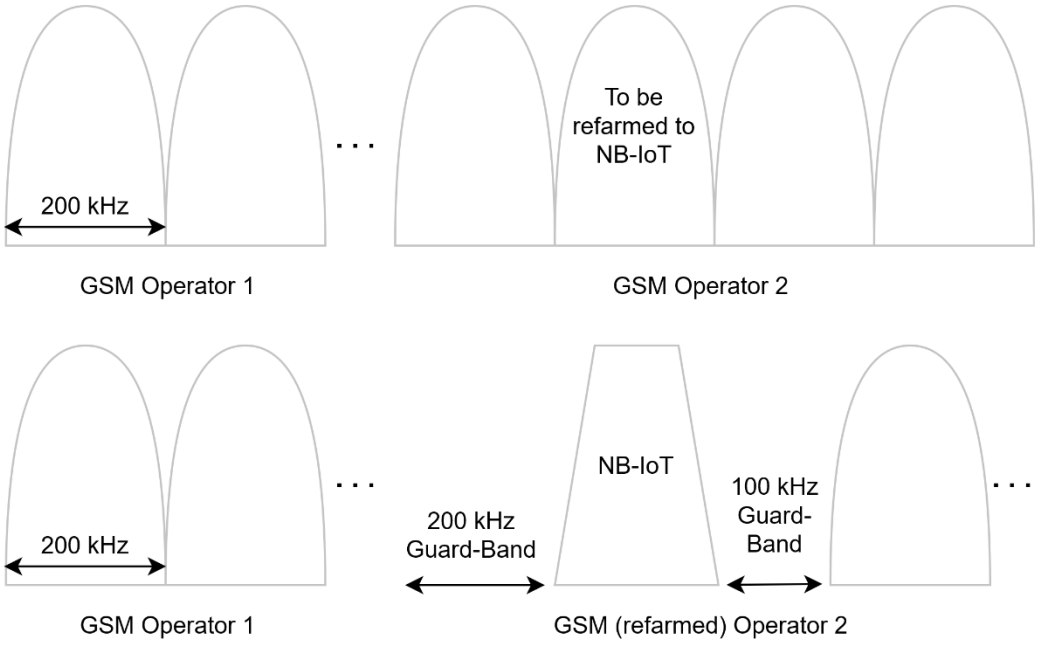

Figure 2.24: NB-IoT stand-alone band in GSM refarmed

b) In-Band

An in-band NB-IoT deployment takes place either directly on an LTE PRB or on an LTE-M unused narrowband. The first option depicted in Figure 2.25 on the left consists in substituting a PRB within an arbitrary LTE band with a NB-IoT 180 kHz wide carrier. On the other hand, the second possibility involves LTE-M features previously explained in Chapter 2.3.5. Because LTE-M narrowbands are mode of 6 PRB, they are inserted within LTE carriers. Depending on the LTE BW, LTE-M manages to fill them according to Table 2.3, i.e. 1 narrowband in the 1.4 MHz BW or 16 in the 20 MHz. Since almost in all of the LTE BW there is more than a multiple of 6 PRBs available there are some PRBs left

which do not belong to any narrowband. Therefore, these may be used for an in-band NB-IoT deployment. As a consequence, new restrictions and challenges regarding the coexistence of NB-IoT, LTE-M and LTE raise; these will be described in the next Chapters [17, p. 223].

c) Guard-Band

As the in-band deployment does, also guard-band's refers to LTE. In the scenario presented in Figure 2.25 on the right, NB-IoT reutilized LTE's guard-bands left at the edges of the carrier. Based on the fact that 5% of the LTE BWs 3, 5, 10, 15 and 20 MHz is made of guard-bands on each side, NB-IoT may again insert its 180kHz carrier at the carrier's borders [17, p. 223].

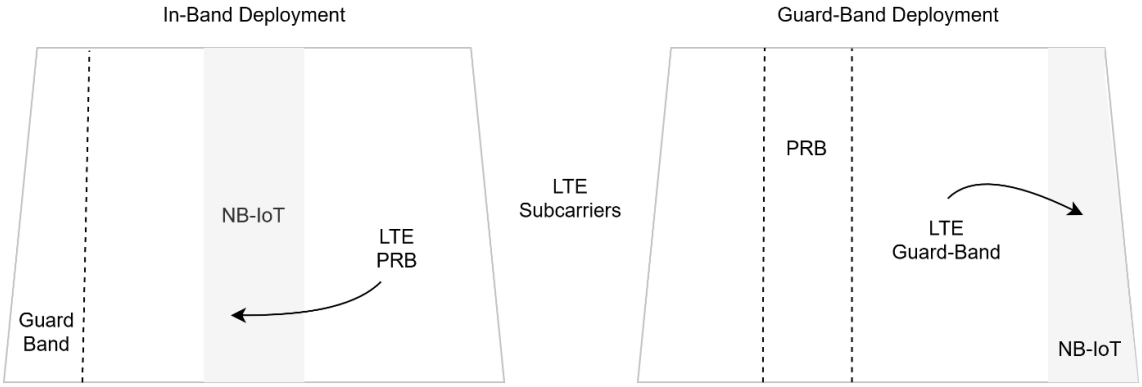

Figure 2.25: NB-IoT In-Band and Guard-band Deployment

2.4.3 LTE and NB-IoT coexistence challenges

NB-IoT deployment in form of in-band and guard-band within the LTE spectrum creates new complexity and possible inconveniences. First of all, 3GPP allows the existence of an NB-IoT PRB next to an LTE's without any guard-band in between. This requirement seems even more ambitious since the system must keep LTE's orthogonality avoiding any Inter Symbol Interference (ISI). Moreover, NB-IoT carriers "share the same time-frequency resource grids as LTE the same way as different LTE physical channels share time-frequency resources" [17, p. 225]. This implies a complete integration not only within the same BW but also with external channels and causes a restricted access to the PRB schema to the NB-IoT resource elements (RE). The Physical Downlink Control Channel (PDCHH) for example avoids NB-IoT of taking the first three REs OFDM symbols in every subframe. The same happens for other LTE physical channels and signals like Channel State Information Reference Signal (CSI-RS) or the Primary Synchronization Signal (PSS).

2.4.4 Cell Selection, Reselection and System Information Acquisition

The process of collecting and exchange information by the modem before entering the connection mode require the same step as described for LTE-M in chapter 2.4.4.

2.4.5 eDRX and Power Saving Mode

As EC-GSM and LTE-M do, also NB-IoT incorporates 3GPP's power saving techniques. The first one is DRX and later its evolved version or eDRX. In this case the functionalities are very similar to LTE-M as described in Chapter 2.3.8 and 2.3.9. The difference is that the maximum cycle period in DRX corresponds to 10.24 s while in eDRX it becomes equal to 2h, 54 min and 46s (one hyperframe). Regarding PSM, the backgrounds remain

the same and Figure 2.23 in Chapter 2.3.10 gives an overview about its typical behavior in the time-domain.

3 Comparisons

This Chapter summarizes the most important concepts presented in Chapter 0 and draws a comparison between LwM2M and MQTT as well as LTE-M and NB-IoT.

3.1 LwM2M and MQTT

Modern IoT protocols were born to extend transport protocols such as TCP and UDP. The requirement was to add a minimum amount of logic allowing devices to communicate adding the least possible overhead. This is one of the reason why MQTT was conceived to be so skeletal: its specification does not define any security mechanism as same goes with the payload which is not imposed [9]. The developer is indeed in charge of designing the application on a higher level. A lot of MQTT projects like Mosquitto for example use TLS as the security level for the encryption and AWS IoT Core gives the possibility of choosing JSON to encode the payload. This results in a vast freedom in regard to the use-case but does not give a guideline for a robust solution. In the end MQTT defines a minimum valuable protocol that is extremely versatile and is reliable for delivering IoT from client to client. Else ways when speaking about LwM2M, the concept of device management protocol fits the best. It locates on top of the so-called IoT messaging protocols such as MQTT and CoAP, by delivering all their features plus well-defined security and data model concepts. Topics like Bootstrap and Objects (Chapters 2.1.3 and 2.1.7) help the user/customer investing a much lower amount of time and resources than taking a lower protocol and defining an appropriate logic for a specific use-case. Therefore, MQTT and LwM2M represent different but at the same time also coexisting protocols: different because they are on two diverse layers and coexisting since LwM2M may also adopt MQTT as the underlying messaging protocol (2.1.1). Nevertheless, it makes sense to compare each other because user may still decide in favor of one of the other: either adapting and creating an own solution with device management features starting from MQTT or directly adopting a ready one like LwM2M and do minimal changes. Below the text highlights the main differences between the two protocols. Table 3.1 delivers a first overview on the two IoT protocols.

Feature	LwM2M	MQTT
Transport Protocol	CoAP, MQTT, SMS, NIDD	TCP, SSL, WebSocket
Communication Type	Server-Client	Client-Broker-Client
Data Format	TLV, CBOR, JSON, SenML JSON, plain text etc.	Out-of-Scope
Security	DTLS, TLS	Out-of-Scope
Protocol Type	Device Management	Messaging
Reliability	CoAP CON and NON	QoS 0,1,2
Data Model	Objects and Resources	Out-of-Scope

Table 3.1: LwM2M vs MQTT

3.1.1 Protocol Stack

MQTT is relying on TCP/IP or WebSocket as its underlining protocol being effectively a connection-oriented protocol. This choice is then abstracted through the Quality of Service (QoS) levels: QoS 0 as a fire and forget method to use as less packets as possible when publishing a message, QoS 1 delivering an extra confirmation of success when sending a message but without avoiding possible duplicates and QoS 2 delivering both consistency when publishing and absence of duplicates at the same time. This is an expedient to decide different level of complexities and overhead based on the use-case. LwM2M on the other hand, because it primarily bases on CoAP and UDP, has less freedom in adapting the QoS. CoAP gives the possibility of choosing between a confirmable (CON) and a non-confirmable (NON) message. The first is the more reliable between the two since the server either sends back an ACK if the CON was successfully delivered or an RST if there were problem in handling the message. The NON represents the pure UDP connectionless message which does not require any confirmation. This is optimal on a power consumption and overhead perspective and is the most efficient way of communicating for constrained devices. When dealing with high payloads, due to firmware updates for example, CoAP and therefore LwM2M splits the content of one single message into multiple packets. This operation is called Block-Wise Transfer (BWT). In contrast to MQTT, where TCP performs the segmenting and resequencing out-of-the-box, CoAP can rely on UDP just for fragmentation [21]. Therefore, CoAP becomes responsible for dividing the packets, as soon as the block size reaches 1024 bytes, and makes sure that they arrive at destination. The BWT is the solution of this challenge and uses CON messages to acknowledge the different chunks.

3.1.2 Security

One of LwM2M's main feature is the Bootstrap Server. It acts as key distribution center and delivers authentication credentials to the client in order to start communicating with the server. LwM2M also include the possibility of bypassing the Bootstrap Server by performing a factory or smartcard bootstrap that would help reducing the initial connection overhead when exchanging the keys. The specification also reports the supported underlying security protocols. Since this thesis' evaluation focuses on CoAP and UDP when dealing with LwM2M, DTLS is the natural choice for the encryption of messaged in this case. As mentioned in Chapter 2.1.8 the security object specifies the DTLS mode used: pre-shared key (PSK), raw public key (RPK), certificates (Cert), no security (noSec) and certificates with enrolment over secure transport (EST). This range of possibilities shows the deep integration between DTLS and LwM2M. Also, the DTLS 1.2 Connection ID (CID) is essential for sleeping devices (Chapter 2.1.9) and proves one more time how the device management protocol relies and specifies recommended security solutions. A client may also use Access Control Lists (ACL) to prevent servers to read/write/execute certain objects and resources. MQTT, on the other hand decides to abstract any security mechanism limiting itself to just suggesting the most suitable ones: SSL and TLS. Related open-source projects like Eclipse Mosquitto for example, include TLS help functions in order to establish an authentication either via PSK or Cert with the ciphers defined by OpenSSL[1]. Nevertheless, the specification only defines an authentication method via username and password which is however transmitted in plain text between broker and client and in any way encrypted. Only MQTT version 5 introduces a tokenized way of providing encryption. Finally, the broker may consent or prohibit clients of publishing and subscribing to certain topics.

[1] Toolkit for the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols

3.1.3 Formats and Data Model

As mentioned in the introduction to this Chapter the MQTT specification does not define any specific format for the protocol. One of the most used solution adopted by open-source projects is sending the payload as plain text or JSON for further data management or storing. This may not be the best choice in terms of message size: TLV, SenML JSON and CBOR are preferred to maintain a limited payload dimension. LwM2M supports all of these because as a device management protocol tends to deliver ready-to-use data for further applications in the back-end and at the same time tends to keep the messages as lightweight as possible. Data Model is another crucial aspect for back-end solutions. The more organized the data arrive the better may DBs and data management platforms handle those. MQTT leaves this task to the user by not providing specific data structure to payloads. The pros of this include freedom in modelling resources. LwM2M has contrarily the opposite approach: depending on the use-case the user must choose one more or more given objects and resources (see Chapter 2.1.7). OMA provides a registry with all of the different possibilities. The benefit here is a tidy and minimalistic way to define use-cases reducing once again the deployment complexity.

3.1.4 Deployment and Scalability

Deploying a LwM2M topology means having to deal at least with two entities: one server and one client. The active part is the server which requests and retrieves data from the client. This solution helps on one side reducing the computational resources since only two actors are at least involved, but on the other causes unwanted data traffic. This is the case for example when a client just wants to communicate sensor data. Until LwM2M version 1.1 the server would have to request an observe operation, before receiving the actual sensor data through notifications, causing a bidirectional flow. Since version 1.2, the specification also include the unidirectional Send operation by the client (for more information please refer to Chapter 2.1.6). Scalability may be also a designing issue. Although some open-source LwM2M projects like Leshan (see next section), include an example of servers' clusters for deployment in scalable environments with thousands of devices, load-balancing the incoming information remain a challenge. Many load-balancers do not support UDP indeed. One of the few possibility is called Linux Virtual Server (LVS). Another solution for LwM2M would be using MQTT as the underlined protocol which works with TCP instead of CoAP and UDP.

MQTT on the other hand has a minimum constellation of 2 clients and one broker. There are also solutions which include just one client and one broker; nevertheless, these become effectively three since the broker back-end would play the role of a subscribed client. Adding one entity means higher complexity as a published message has to reach two endpoints generating at least double as many TCP packages than a simple client-broker scenario. In contrast to LwM2M, clients may publish messages independently to any broker's or other client's requests. This allows a much lighter apparatus when transmitting sensor data. Also, MQTT clients are the one initiating the connection. This solves NAT traversal issues which take place when the server starts the communication.

3.1.5 Industry and Popularity

The IoT market seems to have found in MQTT a preferred approach when dealing with devices. Major IoT device management and cloud platforms like Amazon AWS, Microsoft Azure and Google Cloud support MQTT data exchange. As OMA and the cellular industry are the minds behind it, LwM2M is already part in form of clients of many cellular modems like Quectel's and Nordic Semiconductors'. Speaking about device management

platforms as of May-2021 there is a slow trend in adopting LwM2M. Also, in this case the one involved are cellular providers with a few exceptions: Nokia, Telefonica, Huawei, ARM, Bosch and more. Nevertheless, AWS, Azure and Google Cloud still have not released any information about a possible support. Being the three most popular cloud and IoT platforms they have the ability to guide the trend of a new IoT device management protocol like LwM2M. Regarding open-source projects, LwM2M comes with clients already integrated in cellular modem and microcontrollers as mentioned before. Still on client-side Eclipse maintains two demo projects: Wakaama [22], a C client implementation for constrained devices and Leshan [23], a more generic Java client. Both stick to LwM2M version 1.1. Yet Leshan also include a Server and Bootstrap Server which work with CoAP and DTLS using PSK and Certificates. As MQTT has older roots, there is a larger variety of choices. Some of the open-source clients include: Paho [24] (differentiated in C, C++, Java, Python etc. libraries) and Mosquitto (C library) [25] by Eclipse. Mosquitto also implements a C broker. There are also other projects especially in form of a web or smartphone application to simulate and abstract the operation of publishing and subscribing. These are particularly useful for tests and simulations requiring a minimum configuration effort. The quantity of open-source implementation is one of the main benefit when choosing MQTT. Being still recent and less popular in the IoT market, LwM2M still lacks documentation and differentiation especially in programming languages.

3.2 LTE-M and NB-IoT

The reason behind the development of LTE-M and NB-IoT in 3GPP's releases 12 ,13 and 14 are the same: modem and equipment cost reduction, coverage enhancement, improved devices' lifetime, and scaling LTE devices. Both come from the need of creating an LPWAN standard using the existing LTE infrastructure. Using regular LTE for IoT constrained devices would have led to worse battery-life performances compared to unlicensed access technologies like Sigfox and LoRaWAN as well as high device costs to support the same hardware requirements as smartphones have. Below the Chapter presents the main points of comparison between LTE-M and NB-IoT. Table 3.2 gives a simple overview of the two cIoT standards.

Feature	LTE-M	NB-IoT
Deployment	In-band	In-band, stand-alone and guard-band
Bandwidth	5 and 1.4 MHz	180 kHz
Duplex Mode	Full and Half TDD and FDD	Half FDD
Device's An-tenna	1	1
Mobility	Handover also in Connected Mode	Handover just in Idle Mode
PSM and eDRX	Supported	Supported
Transmit Power	20-23 dBm	20-23 dBm

Table 3.2: LTE-M vs NB-IoT

3.2.1 Deployment

LTE-M comes with two kinds of narrowband deployments: 1.4 MHz for Cat M1 devices and 5 MHz for Cat M2 and non-BL. Since Cat M1 devices are sometimes directly referred to the LTE-M standard and represent the most popular hardware available within this access technology, this comparison will just take in account the 1.4 MHz BW. LTE has the following system bandwidth available: 1.4, 3, 5, 10, 15 and 20 MHz. These are used to receive the LTE-M 1.4 MHz narrowband deployment. Operators have the chance to insert for example 8 narrowbands in the LTE 10 MHz BW or 12 in the 15 MHz (for more information please refer to Chapter 2.3.4). If in this operation there are some bandwidths left like in the previously mentioned 10 and 15 MHz cases, these intervals may be used for LTE ordinary operations. NB-IoT has a different and more variable approach regarding deployment. The stand-alone method takes place in GSM bands and is about refarming a part of their spectra. In GSM this represent substituting a 200 kHz band in a 180 kHz NB-IoT band which would cause a loss of 20 kHz. In a long-term perspective this could be mitigated by the introduction of 5G NR on refarmed GSM spectra and its integration with NB-IoT. The In-Band deployment refers on the other hand to LTE or LTE-M. Here the 180kHz narrowband is either placed on one LTE PRB or on the spare PRBs deriving from LTE-M deployment as explained in Chapter 2.4.2. The final NB-IoT approach is using LTE's guard-bands or at the edge of the carrier's boarders. From this summary it appears that LTE-M chooses a more conservative and effective approach by just relying on LTE carrier frequencies while NB-IoT also uses refarmed GSM bands as well as LTE's guard-bands. The narrowbands of the two cellular standards are also different: 1.4 MHz for LTE-M and 180 kHz for NB-IoT. This translates in different UL and DL performance as well as hardware requirements and therefore complexity and costs. This topics will be part of the next sections.

3.2.2 Data Rates and Cost Reductions

The cost reduction goals induced LTE-M to include in Cat-0 devices a single antenna along with a reduced maximum power. Direct consequences were the introduction of half-duplex operations and reduced BW (20 MHz) as explained before. All these decisions caused a smaller data peak rate reaching between 5 and 10 Mbps. The objective of Cat-M1 was to make these requirements even more stringent by lowering the BW to 1.4 MHz, data rate to 1 Mbps and power class between 20 and 23 dBm [17, pp. 137-139]. FDD and TDD are part of LTE-M. It is also important to remark the fact that both full-duplex and half-duplex may operate with LTE-M, but this depends in the end whether or not the device has a double oscillator. NB-IoT has even tighter requirements in terms of device complexity and costs. The system BW corresponds to 180 kHz and base-band operations only include one synchronization sequence with a lower sampling rate of 240 kHz in contrast to LTE. This is fundamental to reach memory consumption and processing complexity [26, p. 13]. Also, the choice of FEC like TBCC rather than turbo codes delivers a lower overhead. Lower order modulation schemes and only-half-duplex operations (unlike in LTE-M) reduce data rates. Finally, NB-IoT devices use on-chip PA which is beneficial since like in LTE-M the transmit power level is limited to 20-23 dBm. Overall, NB-IoT and LTE-M have the same cost reduction goals. What is different is the concept of budget-cap and the methodology used. On one side LTE-M keeps minimum requirements still allowing a peak data rate of 1 Mbps by using higher modulation schemes and more complex FECs like in LTE and it has reduced BW (1.4 MHz) but still much higher than NB-IoT (180 kHz). On the other, NB-IoT accept a lower data rate but reduces devices costs by just relying on half-duplex, on-chip PA, lower modulation schemes like QPSK and BPSK and less complex FECs [17, pp. 220-221].

3.2.3 Coverage

Despite the strict cost reductions, both access technologies aim at delivering robust coverage performances. NB-IoT comes with three different coverage extensions (CE) levels: CE level 0 representing the base level as well as the more advanced CE 1 and 2. Those three operate according the actual signal conditions, namely RSRP as well as Carrier to Interference and Noise Ratio (CINR). In order to increase the coverage, these three levels choose a specific number of downlink and uplink repetitions, as defined by the network operator [27]. As mentioned in the previous section, NB-IoT relies on QPSK and BPSK in the UL channel. Here the uplink modulator maintains a close to constant envelope, hence the PA works at the maximum transmitting power with no need of power back-offs [17, p. 221]. This is a big advantage for a robust coverage. LTE-M follows a similar approach by defining CE levels according to the number of repetitions: CE mode A is obligatory on Cat-M1 and -M2 devices and allows up to 32 subframe repetitions while mode B is not mandatory and supports up to 2048 repetitions. These modes are vital to reach the 20 dBm coverage target. As this is the same goal as in NB-IoT, both standards apply similar and comparable ways to reach acceptable coverage for different use-cases. Nevertheless, LTE-M can also apply frequency-hopping making more suitable to frequency diversity [17, p. 346].

3.2.4 Handover

LTE-M maintains the same procedures as LTE for cell-handovers. It is therefore suitable for use-cases where the devices are moving performing one ore multiple cell changes with similar QoS. For NB-IoT this is not the case. In fact, in idle mode the modem has to reselect the cell. During connected mode, this is not possible.

3.2.5 Power Saving Techniques

As explained in Chapters 2.3.8, 2.3.10 and 2.4.5, NB-IoT and LTE-M include the same techniques to save battery-life in idle mode. eDRX and PSM are highly adopted in various kind of cIoT modems and networks.

4 Experimental Setup

This Chapter explains the hardware and software setup used for the measurements. It first deals with the devices used to send and transmit data: modem, extension board and regular board. Secondly, this section presents the power analyzer used to determine the device's energy consumption during data communication as well as the software to capture the packets on server-side. Thirdly, it gives an overview about the hardware measurement setup used for the experiments. Finally, it provides a description of the software and open-source repositories adapted for the test environment.

4.1 Modem

The Quectel BG96 is a popular all-in-one LTE Cat M1, Cat NB1 and EGPRS module. It can be easily integrated with other hardware thanks to industry-standard interfaces like USB, UART, I2C and has compatible drivers for Windows, Linux, and Android. It supports the following LTE-M and NB-IoT FDD bands: 1, 2, 3, 4, 5, 8, 12, 13, 18, 19, 20, 25, 26, 28 and 39 (TDD just for Cat M1 devices). In Germany, for example, Deutsche Telekom utilizes bands 20 and 3 for LTE-M and 8 for NB-IoT while Vodafone 20 for NB-IoT. This is the trend for other operators in the EU too. In USA one can find bands 2, 4 and 12 operated for example by AT&T and T-Mobile US for both NB-IoT and LTE-M [28]. The BG96 does not allow the LTE-M maximum downlink and uplink speed of 1 Mbps. It reaches a maximum of 375 kbps for both DL and UL. Using NB-IoT the values are as expected lower: 32 kbps (DL) and 70 kbps (UL). As imposed in the cIoT standards the maximum transmit power is 23 dBm. Table 4.1 reports some typical consumptions values that may become relevant in the measurement section [29].

State	LTE Cat M1	LTE Cat NB1
Power Saving Mode (PSM)	10 µA	10 µA
Idle State (DRX=1.28s and eDRX=40.96s)	15 mA	15 mA
Sleep State (DRX=1.28s)	1.5 mA	1.96 mA
Sleep State (eDRX=40.96s)	1.2 mA	1.1 mA
Connected Mode (Avg.) @23 dBm	205 mA	223 mA

Table 4.1: Quectel BG96 Consumptions [29]

The module also supports a variety of protocols. These are to be configured via AT Commands (see Chapter 4.3) and include: PPP, TCP, UDP, SSL, TSL, FTP(S), HTTP(S), MQTT, CoAP and LwM2M [29]. The BG96 comes in fact with integrated MQTT and LwM2M clients. This is particularly useful to exploit the modem's computational resources instead of relying on external boards that just use the modem as a gateway to the cellular network. The benefit is also a ready-to-work client without the need of configuring and installing a new one on an external board. Nevertheless, having tested the LwM2M client on the BG96, the objects and resource available appeared to be insufficient for the scope of this research and MQTT and LwM2M clients on a regular board were used (Chapter 4.7).

Figure 4.1: Quectel BG96 [29]

4.2 SIM Card and Operator

The experiments rely on the connectivity provided by the MVNO 1NCE and its SIM cards. Typical cellular operators still do not have the motivation and freedom of expanding the IoT market making it accessible to everyone in a simple manner. They offer IoT plans only to business companies. 1NCE on the other hand has the mission to make cellular connectivity open to everyone. By offering an IoT flat rate it supports every kind of user: from the most expert costumers and companies to the beginners developing their own IoT projects. As a Mobile Virtual Network Operator (MVNO) it offers both M2M regular and electronic SIM Cards able to support GSM, UMTS, LTE, NB-IoT and LTE-M. It also allows a 10 years data retention and 500000 read/write cycles. There are also other services available: among others a Rest API for Connectivity management, OpenVPN to establish a secure connection between 1NCE's network and the own application server, and Connectivity Suite to enable plug-and-play devices communication directly with AWS services [30]. Moreover, there are vast number of countries included in 1NCE roaming agreements. As of mid-2021 the majority of EU countries and China appear in the NB-IoT roaming agreements while US and Canada in the LTE-M. As this is slowly becoming popular also within Europe, the majority of the operators still does not apply roaming restrictions within this area.

4.3 AT Commands

There a special set of instructions in order to configurate a cellular modem. These were invented by Dennis Hayes for its Hayes Smartmodem 3000 baud modem in 1981 [31]. AT stands for Attention and there are different types of requests as described Table 4.2. "The AT Commands Set implemented by BG96 is a combination of *3GPP TS 27.007*, *3GPP TS 27.005* and *ITU-T recommendation V.25ter* as well as the AT Commands developed by Quectel" [32, p. 8]. Theoretically all of the BG96 interfaces allow to send and wait for AT commands requests and responses. Nevertheless, as the work will describe in section 4.4 the extension board only supports UART and USB. Both choices are reliable and although the USB baud rate is higher this normally does not play any role in the modem configuration.

Test Command	AT+<x>=?	This command returns the list of parameters and value ranges set by the corresponding Write Command or internal processes.
Read Command	AT+<x>?	This command returns the currently set value of the parameter or parameters.
Write Command	AT+<x>=<…>	This command sets the user-definable parameter values
Execution Command	AT+<x>	This command reads non-variable parameters affected by internal processes in the UE

Table 4.2: Types of AT Commands and Responses [32, p. 9]

There are various procedures to establish a cIoT connection. Listing 3.1 reports one procedure example that helps scanning LTE-M's Radio Access Technology (RAT). After that the user may choose a preferred LTE band (in this case 20 as the connection took place in Slovakia) and consequently set up the APN, 1NCE's in this case. The APN field is mandatory and allows the authentication in the default operator network as well as roaming authorization in the local one. Performing a manual registration inserting the provider's code (in this case Orange SK, 23101) helps then reducing energy consumption in the attaching and connection phase, especially in roaming environments (please find a list of more cellular operators and their code in appendix LTE-M and NB-IoT Operators). The same is true when setting the band: limiting the scanning helps saving battery power but could be limiting the connectivity in dynamic scenarios. Because this version of the BG96 does not allow the preferred operator list AT command (`AT+CPOL`) there are some cases where choosing manually the network operator is the only choice. When powering on, the modem tries to attach to the wrong cellular provider which 1NCE does not provide the roaming agreement with. This is the case in Italy where both Vodafone Italia and Telecom Italia offer NB-IoT frequencies on band 20 (see Table 6.4 in the appendix). Here the BG96, presumably because of the signal condition in a specific location, only tries to attach to Telecom Italia and ignores Vodafone Italia which 1NCE has the Italian roaming contract with. It is in some scenarios therefore important to manually set up operator and frequency band. Finally, the command `AT+QNWINFO` in Listing 3.1 outputs RAT, operator, band, and channel in the following form:

```
+QNWINFO: "CAT-M1","23101","LTE BAND 20",6200
```

which provides a summary of the chosen settings. Please find in appendix B. the AT Commands procedures for connecting the modem to NB-IoT.

```
// LTE-M scan mode configuration
AT+QCFG="nwscanseq",02,1
AT+QCFG="nwscanmode",3,1
AT+QCFG="iotopmode",0,1

// Set LTE-M Bands to LTE Band 20
AT+QCFG="band",0,80000,0

// Define PDP context using 1NCE APN
AT+CGDCONT=1,"IP","iot.1nce.net",,

// Manual Registration to Orange SK
AT+COPS=1,2,"23101",8

// Retrieve network and operator info
```

Listing 4.1: LTE-M Modem connection procedure through AT Commands

4.4 Extension Board

In order to enhance the services and tools available to a cellular modem like the Quectel BG96, the embedded system industry manufactures extension boards. These are integrating modems and add extra functionalities like sensors, own firmware, and interfacing libraries. The Sixfab Cellular IoT HAT is an extension board (Figure 4.3) which contains the Quectel modem, a SIM Card slot, and provides a micro-USB interface as well as 40 GPIO pins thought among others for 3.3 V and 5.5 V input voltage, ground, I2C and UART. All data pins work with 3.3 V reference. There are also three different LEDs: Status LED, which blinks when the board is powered, User LED, to show the power regulator status, and Netlight LED, indicating when data is transmitted or received. As pointed out in Chapter 4.1 the BG96 supports GNSS of various kind: GPS, GLONASS, BeiDou/Compass, Galileo and QZSS. The Cellular IoT HAT supports this functionality by integrating a passive GPS antenna circuit by default. The shield has in fact a GNSS antenna socket as well as an equivalent LTE one. The dual antenna is called Pulse Electronics Gemini Series and is depicted in Figure 4.2. It operates in bands B1-B23, B25-B29, B33-B42 suitable for EU and Asia. The supported frequencies are 1559-1610MHz. It is also foldable for tight spaces and represents a reasonable choice for constrained cellular devices where the complexity must be balanced with the manufacturing costs. Moreover, as explained before, the Cellular IoT HAT offers slots for 40 GPIO pins as visible from Figure 4.3. In fact, Sixfab designed this device as an add-on for the Raspberry Pi (RPi) which also hosts the same number and kind of pins. By stacking the IoT HAT on top of the RPi, the BG96 receives the necessary input voltage via GPIO and can at the same time deliver the UART and I2C interfaces via pins. From the figure in appendix C. one may have a deeper look into the GPIO and their role. Another important aspect about the shield is the Point-to-Point (PPP) link coming from the BG96. This is offered either via the micro-USB interface which can be exploited with a micro-USB to USB cable or via UART. This solution makes the board flexible to guarantee a cellular connection to any kind of device, from personal computers to other embedded systems. In case the PPP connection is exploited via UART however, the DL and UL speed decrease. The extension board also comes with a Python library to facilitate the use of AT Commands as well as control hardware components like LEDs

and GPIOs [33]. Nevertheless, as of May 2021 the library is still limited and does not allow hosting java and C projects for LwM2M or MQTT. Since this thesis' primary objective is the device and network performance analysis using LwM2M and MQTT open-source projects, the Python library becomes irrelevant for the scope of this work.

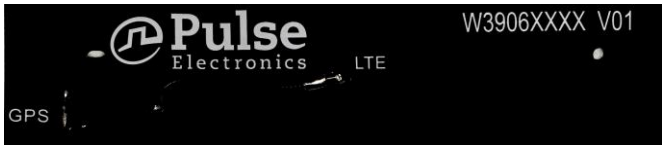

Figure 4.2: Multi-Band Antenna

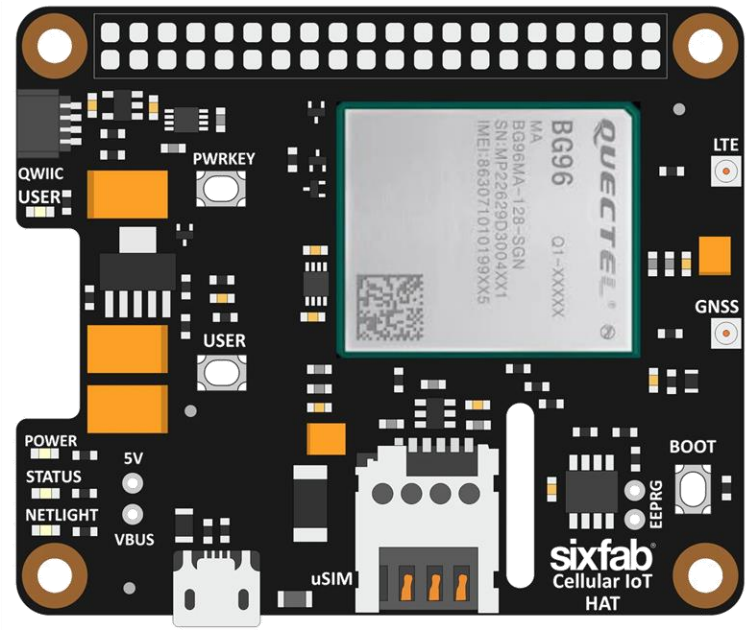

Figure 4.3: Sixfab Cellular IoT HAT [33]

4.5 Power Analyzer

When evaluating IoT protocols within constrained devices the energy consumption analysis becomes relevant. Yet it is sometimes difficult to measure devices in real-life conditions. Mobility in relation to energy consumption for example may be difficult to analyze with standards tools. The Qoitech Otii Arc is a compact power analyzer suitable for measurement in any location. Its dimensions and weight are 10.9cm x 14.4cm x 4.4 cm for 450g [34]. It is also a power supplier and provides a constant current and voltage between 500mV and 5V. When powering the Otii Arc only via USB the maximum voltage delivered does not exceed 3.75V otherwise it reaches 5V. Because the extension board requires a voltage in range 3.3 to 5V, all the measurements were performed with 3.75V to avoid the need of other power sources other than the laptop USB. This is especially important when measuring in mobility scenarios. "The Arc has an accuracy of ±(1% + 0.5μA) and a sample rate of 4kHz in the lower current region and 1kHz when the current is larger than 19mA" [26, p. 21]. This feature makes it particularly suitable when detecting the current or energy consumed in PSM which for the IoT HAT is supposed be around 10μA [33]. On the other hand, the voltage accuracy reaches ± (0.1% + 1.5mV) with a sample rate of 1ksps. For more details refer to Appendix D. As illustrated in Figure 4.4, the Otii's front part has 14 pins and two banana female connectors for + and - voltage input. Within the available pins there are the UART TX and RX. These becomes useful

when syncing logs to current behaviors. It is therefore possible to configure own scripts, like the LwM2M and MQTT clients for the scope of this work, to utilize the UART port for logs (please find more details in Chapter 4.7). As depicted in Figure 4.5 within the Otii's software (which is freely available [34]), the user may then select parts of the captured time-current graph and receive automatically the related logs coming from the UART ports highlighted. Other than the UART window, the software application comes with the possibility of comparing different recordings; this makes it easy to recognize differences between configurations.

Figure 4.4: Qoitech Otii Arch [34]

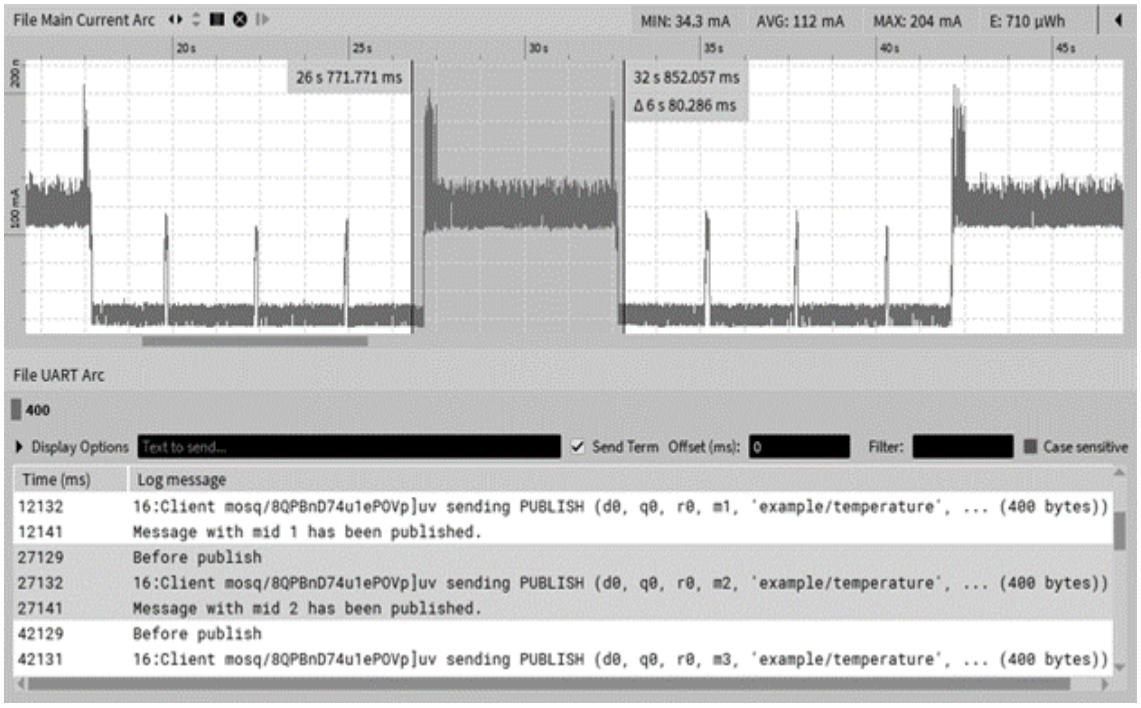

Figure 4.5: Screenshot from the Otii's software showing a highlighted part of the graph synced with the corresponding logs

4.6 Packet Analyzer

Wireshark is a widely-used packet and network protocol analyzer. It allows inspecting in depth incoming and outgoing packets. It also gives the possibility to filter a particular port, IP address or interface to monitor outgoing or ingoing packets from. All of the most common protocols are supported. Focusing on IoT messaging and device management protocols we find: MQTT, CoAP, AMPQ, XMPP and MQTT-SN. Regarding LwM2M, Wireshark as of May 2021 just supports Lightweight M2M TLV. From the documentation this seems to be an initial version that only support payload with a TLV encoding. It can recognize predefined and connectivity resources along with their data types. Other protocols relevant for this work are the secure TCP and UDP variants: TLS and DTLS.

However, when retrieving this packets, their payload appears as expected encrypted. Wireshark also provides a terminal based version called Tshark. It is a lightweight network analyzer running on desktop-less operating systems. This is ideal when for example connecting to a remote VM without desktop access. Tshark is analogue to tcpdump when running without options. A typical command to capture packets arriving at a LwM2M server on port 5684 may look like this:

```
tshark -i ens5 -f "udp port 5684" -w server_dtls_.pcap
```

This command ensures capturing UDP packets flowing on interface ens5 and saving the session results in a .pcap file. This is then transferred to a desktop computer for a deeper analysis thanks to Wireshark. The file conserves in fact all of the capture's details, that on a desktop-less machine would be harder to analyze.

4.7 Measurement Setup

Figure 4.6 shows the hardware and measurements setup which consists of three major components:

- Power Analyzer - Qoitech Otii Arc
- Extension Board with Cellular Modem and Antenna – Sixfab IoT HAT with Quectel BG96
- Regular Board – Raspberry Pi 3 Model B

Besides the power analyzer and the extension board already presented in Chapters 4.5 and 4.4, the regular board delivers the software clients for the experiment. The Raspberry Pi is versatile in this sense since it hosts Raspbian, a Linux distribution optimized for the RPi. This allows the board to be flexible in supporting all of the majority open-source projects without constraints related to programming languages or framework environments. The extension board does not provide indeed the best solution in this matter since it only offers a Python library for interfacing the modem. The best solution for the scope of this work is therefore to exploit the extension board's PPP interface to transport the data packet from the RPi towards the IoT HAT and subsequently to the modem. The PPP does not add any noticeable delay and allows the BG96 to just send out the data without adding any extra computational effort using the integrated MQTT and LwM2M clients (for more information about the integrated libraries please refer to Chapter 4.1). As Figure 4.6 illustrates the USB to mini-USB cable (1) between extension board (4) (on the bottom) and regular board (5) (on the left) delivers the PPP link. The power analyzer (2) (on the top) powers the extension board (4) with 3.75 V and a constant current of 907 mA. At the same time, it gathers information about the energy consumption and returns a minimum, maximum, and average current as well as the energy consumption for the selected time-frame. The banana connectors on the right of the Otii's frontal part (3) are inserted into the 5 V and digital ground GPIO pins (6) of the IoT HAT. As described in Chapter 4.5 the power analyzer offers UART TX and RX GPIO male and a digital ground pin on the front-left (7). These are cabled with the Raspberry Pi's UART RX, TX and digital ground respectively (8). The LwM2M and MQTT client on the regular board send constant logs via UART. These appears then on the Otii application software which allows to configure the required baud rate. Finally, Raspberry Pi and Otii Arc are both powered by the laptop's USB ports.

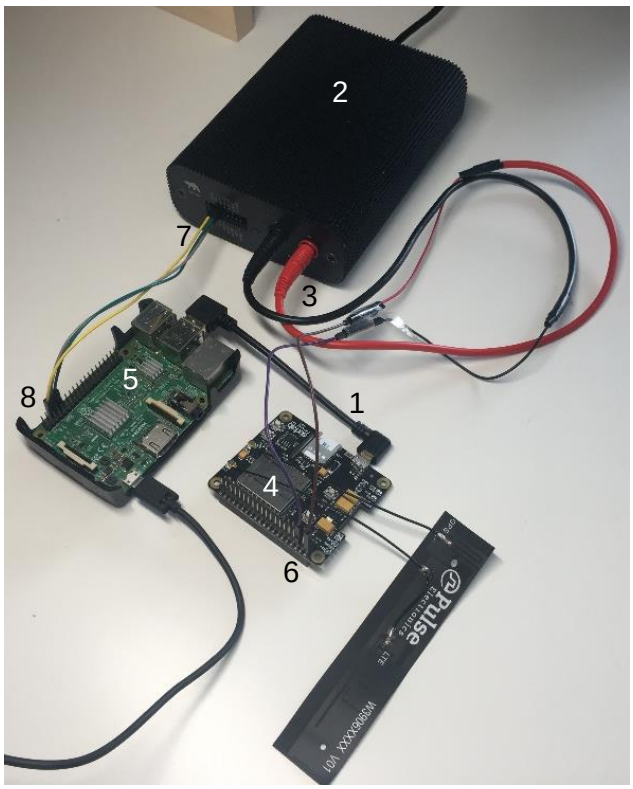

Figure 4.6: Measurement Setup

4.8 Software

This section focuses on the software component used for the experiments. As the development of ad-hoc clients and servers is out of the thesis' scope, the implementation is about adapting open-source repositories. Both LwM2M and MQTT are supported by the Eclipse Foundation which makes free projects available on Github [35]. Out of the available projects, Mosquitto is chosen as implementation of MQTT. Regarding LwM2M Eclipse has developed Leshan and Wakaama, as explained in Chapter 3.1.5. From these, the more advanced and complete is Leshan, which also has high degree of freedom when customizing the solution with extra parameters and variables. The following part of this Chapter dives into Leshan and Mosquitto, the chosen open-source repositories available for LwM2M and MQTT.

4.8.1 Leshan

The Leshan project contains a LwM2M Client, Sever and Bootstrap Server implementations. Each of them comes with development guidelines as well as demo scripts for testing purposes but are not ready for production cases. The thesis takes in account both Client and Server parts describing and adapting them for the experiments. The Bootstrap Server is however not part of the scope. This is because an initial bootstrap phase before the client-server handshake would influence the comparison with MQTT enormously. As a trade-off between energy consumption/delay and security, the experimental setup prefers a factory bootstrap simulation providing to client and server the necessary credentials upon connection. Moreover, the 1.x Leshan version of the repository reflects LwM2M version 1.0.2 while the 2.x LwM2M version 1.1.1. There is still neither short- not long-term plan to include LwM2M version 1.2 within the project.

a) Server

The Leshan server aims at implementing the LwM2M regular Server. It includes a demo Server running on a remote VM. It supports full usage of DTLS through either PSK or X509 certificates. The DTLS Connection ID (CID) is also part of the code and may be: activated, activated specifying the CID length in bytes, activated but without CID generation for the foreign peer or deactivated (for more information about CID please refer to Chapter 2.1.9). When starting the server, it is possible to specify a local webport where to execute the UI. Figure 4.7 shows the server web interface where to manage the client as well as configure the server authentication method. In the client tab it also possible to specify the response timeout when waiting for a response as well as to decide a payload encoding choosing one between the following formats: TLV, plain Text, JSON, SenML JSON, Opaque, CBOR and SenML CBOR. The security tab allows to read the server's certificate and public key as well as to configure the client's credentials. When launching the client via terminal, flags may help setting up the right connection. These options are reported in appendix E. .

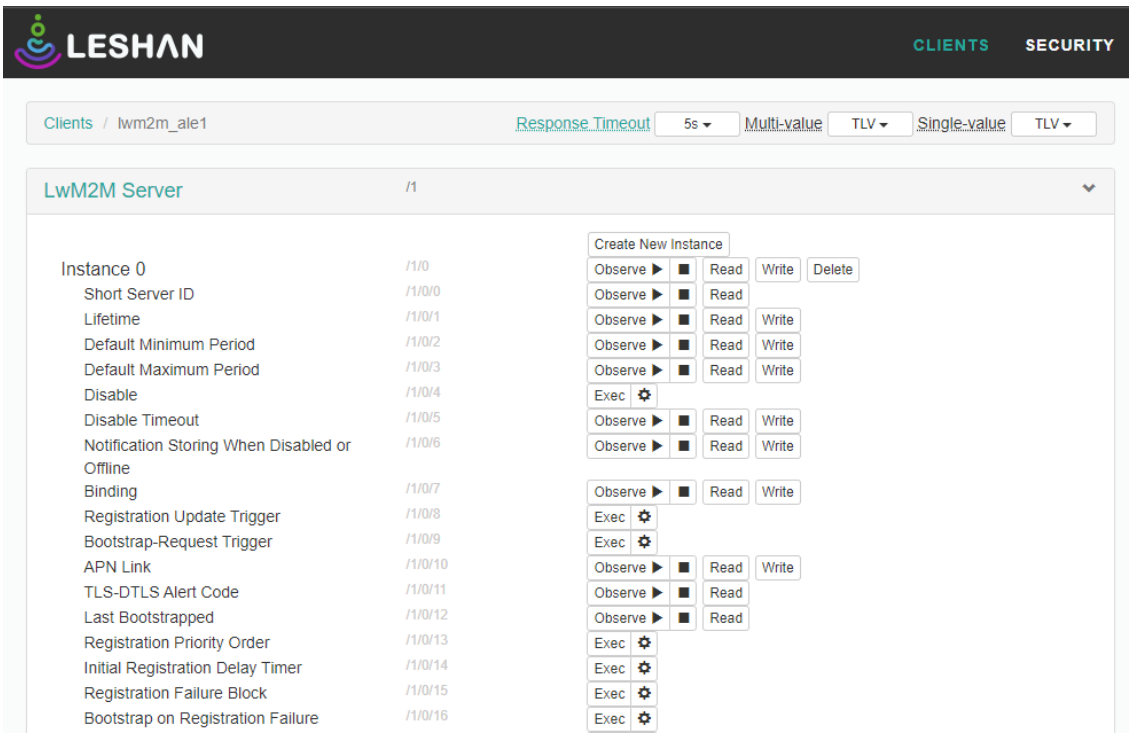

Figure 4.7: Leshan Server Web UI

b) Client

Like the LwM2M Server, Leshan includes a LwM2M Client demo. Other than the DTLS and CID additional options also present in the server, the client contains among others: reconnection/re-handshake on update operation and re-handshake instead of session resume. This choices would fit into a strong security concept where the client performs a full handshake every time instead of an abbreviate one. Since one of the thesis' main aspect is the energy consumption analysis in LPWAN, the experiment does not rely on these flags and includes instead the usage of CID and abbreviate handshakes when needed. There are other configurations related to Californium (library for CoAP and DTLS) that may become relevant when implementing the client. One of those is to choose whether or not the client should use CoAP CON or NON notifications when responding to an observe request from the server. This has potentially a big impact on traffic-over-the-air and energy consumption: the CON includes a confirmation packet by the server which may create longer transmission-durations, traffic, and battery consumptions.

Moreover, in order to perform the measurement with different payloads, the resource /3/0/0 device/manufacturer has been modified. After the change, when the server triggers an observe operation, the notification will include the payload size specified in the added flag “-bs”. For the complete list of addable flags please refer to Table 6.8 in the appendix.

4.8.2 Mosquitto

Once again, the Eclipse Foundation provides an open-source project but this time regarding MQTT. Mosquitto is a message broker implementing the latest protocol specifications, including version 5.0. The official website states that Mosquitto is suitable for every kind of device, either constrained or with extended computational features [36]. The related GitHub page not only contains a C library to configure the MQTT broker but also one for the client. There are other popular client implementations as for example Paho, but these were discarded in the work because of the extensive and complete Mosquitto documentation available for both broker and client.

a) Broker

As the LwM2M Server does, the MQTT Broker receives and sends packet from and to the client. The packets capture, as reported in Chapter Results and Analysis, takes place on broker side as this represents the communication terminal in this experiment. The broker may be customized by adding a configuration file. This is especially necessary when dealing with secure transportation. TLS with PSK requires an identity and a secret key. Listing 3.2 shows the PSK broker configuration file content used for the measurement.

```
port 8883

log_type all
use_identity_as_username true
psk_hint my psk connection
tls_version tlsv1.2
ciphers PSK-AES128-CCM8
psk_file /etc/mosquitto/psk/psk_file.txt
```

Listing 4.2: MQTT Broker configuration file for TLS PSK

After defining the secure port which in MQTT is recommended to be 8883, log types are activated. Then if the variable `use_identity_as_username` is true the PSK identity is used instead of the MQTT regular username. The `psk_hint` enables the PSK within the broker and may be set to any value. After, the file defines the TLS version as well as the TLS ciphers. This should correspond with the one utilized in the LwM2M DTLS PSK configuration to provide fair comparison between the two protocols. Finally, the configuration file links the text file where identity and secure key are stored. Next, Listing 3.3 reports the configuration file for the TLS broker dealing with certificates.

```
port 8884

log_type all
use_identity_as_username true
require_certificate true
cafile /etc/mosquitto/CA/root_cert.pem
keyfile /etc/mosquitto/Server_Cert/keys.pem
certfile /etc/mosquitto/Server_Cert/CA_signed/server.crt
tls_version tlsv1.2
```

Listing 4.3: MQTT Broker configuration file for TLS with Certificates

This time the TLS port is 8884. This choice does not reflect the recommended port in the specification but allows to run both broker versions at the same time on the same machine. The file contains similar variables as in the PSK configuration file. However, this time instead of the identity and password file, the path to the certificates and keys are linked: the certificate authority (CA) key as well as the broker private key and signed certificate. Regarding a non-secure broker, there is no need of defining a specific configuration file. The user may execute `sudo mosquitto -p 1883` and the broker will just start on the non-secure port 1883.

b) Client

The Mosquitto client comes with a well-documented C library. This is actually included directly into the Mosquitto Broker GitHub repository. The experiment as described later in Chapter 5.3 is divided into different phases. There is a different client script for every phase including different configurations within the code:

- Broker IP address and port
- Publish frequency (if needed)
- Quality of Service
- Payload Size (if needed)
- Security TLS type: no security, PSK or certificates
- Path to authentication files (PSK, certificates)
- Supported ciphers
- Supported TLS version
- Lifetime
- Logs

Changing these variables within the experiment means changing use-case and therefore packet and energy consumptions results, as the thesis reports in the next Chapters.

5 Experiments

This Chapter enlightens the experiments' organization and setup. First, the Network Structure gives an overview about the different components enabling the communication, including network properties and providers. Secondly, Preliminary Tests of cellular Networks as well as necessary Hardware and Software delivers a description of eventual issues followed by a potential resolution. Lastly, Experiment Flow and Measurements Details describe scenarios, methods and configurations taken in account within the measurements.

5.1 Network Structure

1NCE deploys its core network within Amazon AWS Virtual Private Cloud (VPC). As depicted in Figure 5.1 the P-GW, which is built on more EC2 machines, is located within the same virtual network as the Application Server does. Another experiment-dedicated EC2 instance runs multiple LwM2M Servers and MQTT Brokers at the same time. These operate on different ports according to the security level. Leshan allows one single PSK- and Certificates-based Server while MQTT requires two different ports for simultaneous execution due to the separated configurations (Chapter 2.2.2b). On the left-side the clients start on a Raspberry Pi whose packets are incapsulated and sent via PPP to the extension board (IoT HAT). At this point the integrated modem (BG96) takes care of the cellular link. Depending on the access technology and APN set, the data reaches the correct S-GW and P-GW (for more information about the cellular network architecture please refer to Chapter 2.3.2). Because 1NCE is a virtual network operator, it does not own cellular infrastructure. Instead it relies on partner mobile operators thanks to roaming agreements. For the measurement the relevant operators are Orange Slovakia and Vodafone Italia.

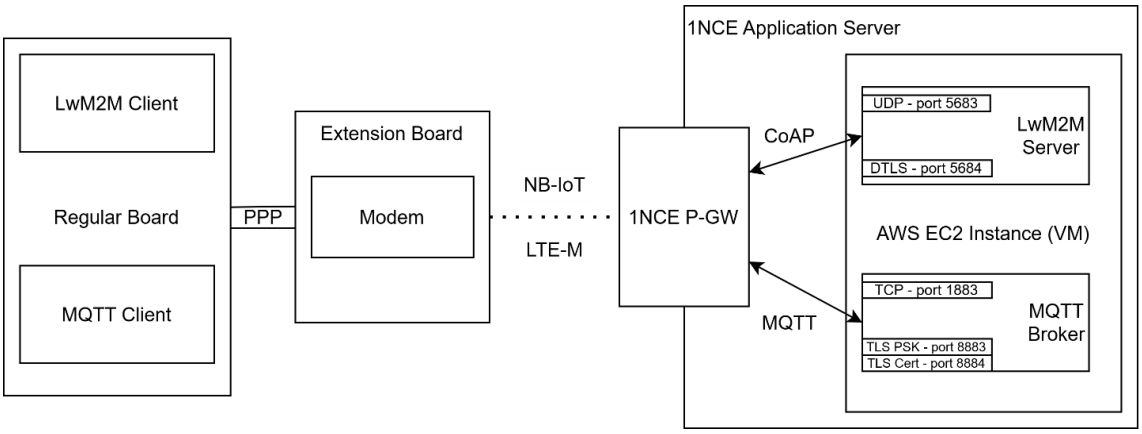

Figure 5.1: Network Structure

5.2 Preliminary Tests

Preliminary tests are needed to understand the behavior of Network, Hardware, and Software. This step is useful to understand and possibly adjust configurations and parameters based on the desired requirements.

5.2.1 Network

The modem handles the communication between edge and cellular network. As described in Chapter 4.3, the AT commands help setting various configuration which include among others: APN, access technology, band, and operator. Once the eNodeB attaches the modem, the connection phase may start (part of this is described in Chapters 2.3.7 and 2.4.4). Once the device is finally connected and does not send or receive any data, DRX or eDRX cycles take place. Figure 5.2 shows the modem's current consumption over time. On the left- and right-side the figure represents the end and the beginning of the idle phase, where the paging periods exist. In the middle the connected mode starts with the data exchange and finishes with the RRC connection release and the successive direct transition into idle mode. As visible from the logs and the synchronized highlighted part on the graph, the data exchange only lasts for the first 583 ms since the beginning of the connected mode. A time-frame of about 5 seconds (see upper time axes) follows. This is called DRX inactivity timer as explained in Chapter 2.3.9. As noticeable, the carrier (in this case Orange Slovakia with its LTE-M network under testing) chooses to enter the idle mode immediately after the DRX inactivity timer skipping the DRX timers (for more information refer to Figure 2.22). The 5 seconds RRC inactivity timer (also present in Vodafone Italia NB-IoT) are not ideal in terms of energy consumption, as described later in Chapter 6. Nevertheless, it turns out to be efficient for those cases where the UE may transmit successive messages with a short time-interval between each other and must stick to stringent delay-requirements. This is beneficial since the UE does not have to re-enter the connected mode which might cause extra-latency. Figure 5.3 depicts the network behavior maintained by Magenta Telekom Austria which applies a DRX inactivity timer of about 95 ms within LTE-M band 20. In this case the DRX timers do not seem to appear too. Finally, Table 5.1 reports the network properties involved in the measurements. Please notice that the used modem can control the DRX cycle time. However, the thesis chooses to not modify this parameter in order to analyze pros and cons of a shorter or longer cycle within the networks.

Access Technology	Provider	(Minimum) DRX	RRC Inactivity Timer
LTE-M	Orange Slovakia	2.56 s	~ 5 s
NB-IoT	Vodafone Italia	10.24 s	~ 5 s

Table 5.1: Network Properties used for the experiments

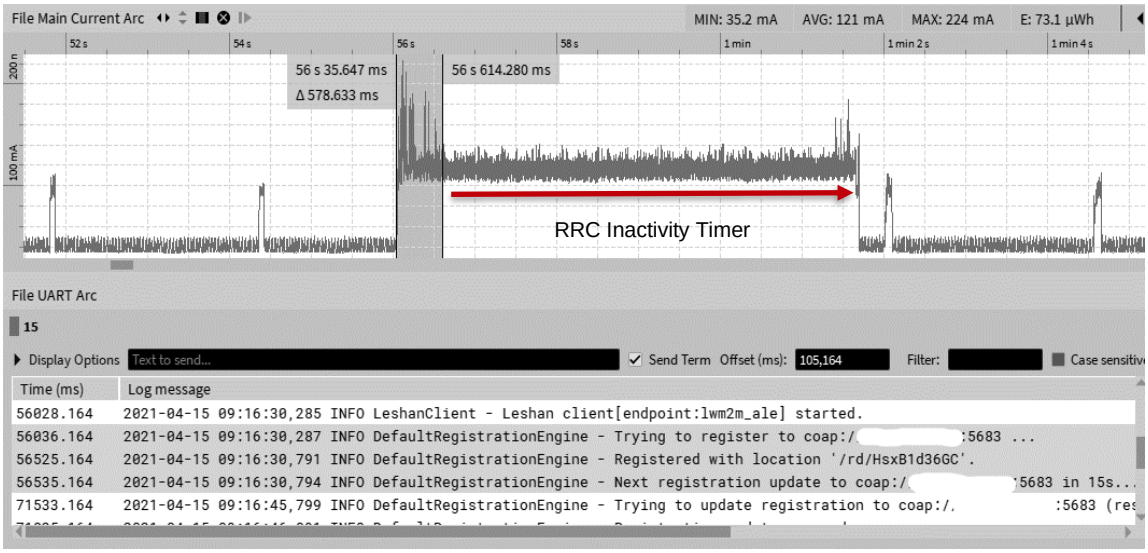

Figure 5.2: LTE-M idle and connection mode with RRC inactivity timer provided by Orange Slovakia

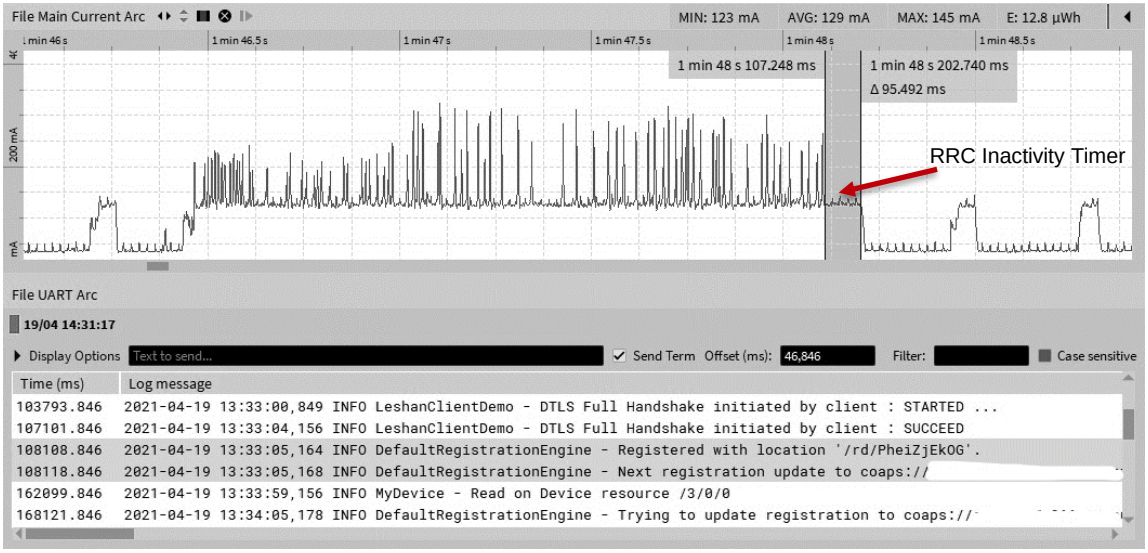

Figure 5.3: RRC inactivity timer provided by Magenta Telekom Austria

5.2.2 Hardware

Besides the BG96, the IoT HAT contains other components that might influence the energy consumption. First of all, the three LEDs play a relevant role. The internal Python library cannot control the way the lights behave. The modem is the only entity able to set the blue Netlight either into blink or static mode; while there are no methods to configure the other two. As monitored during the first tests, modifying the behavior of the Netlight does not effectively decrease the consumed energy. The Sixfab documentation states that the energy consumption values in Table 6.5 "refer to the consumption of only the BG96 module, not the whole circuit at all. With the LEDs and regulation losses, it may rise to 25mA" [33].

Another component which affects the measurement is the USB link dedicated to the PPP tunnel between Raspberry Pi and IoT HAT (extension board) during the experiment. In this case, the PPP connection itself does not play any role in the energy consumption, while contrarily the USB power management increases the used current and creates noise. This is visible from Figure 5.4. It shows on the left the current before inserting the micro-USB into the port and after. Initially the current is constant at about 20 mA (1) and after not only it reaches approximately 37 mA (2) but, after the DRX cycle takes place, it also shows a much noisier behavior (3). The current oscillates between 37 and 43 mA (3). Using the UART link instead of the USB to allow the PPP transport might have avoided this phenomenon. However, when trying to establish a data transmission between Raspberry and IoT HAT, the extension board did not respond. It turned out that to allow this setup, the IoT HAT would also need the Raspberry power source. This was however not possible since during the measurements the voltage must come from the Otii. The outcome is not optimal for measuring the power consumption as it enhances the average energy. Yet it does not affect the comparison between different protocols. The current-increase as well as the noise caused by USB remained constant for both LwM2M and MQTT data-collection during the entire duration of the experiments and did not prevent from detecting data peaks within the graph.

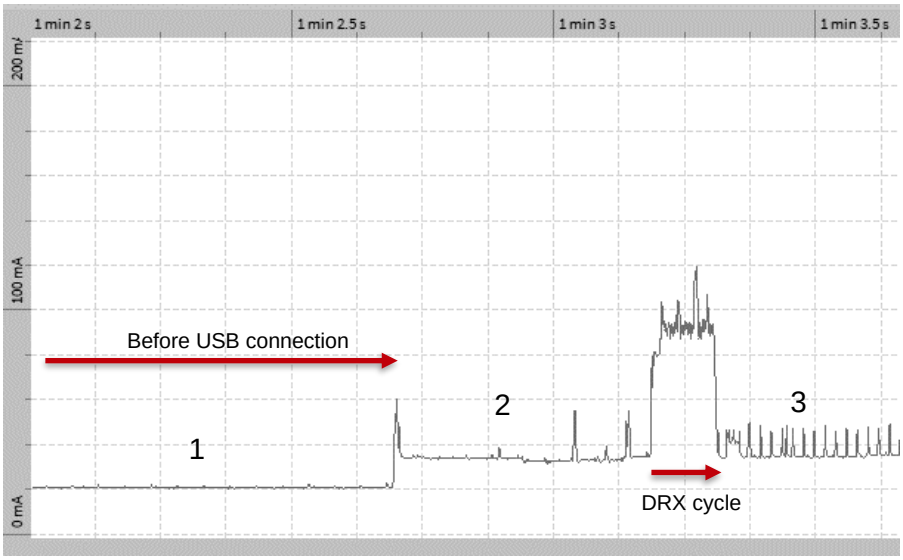

Figure 5.4: Influence of the USB link on the IoT HAT's current consumption

5.2.3 Software

Because the experiments also deal with the packet analysis, the software and its configuration may play a relevant role in terms of efficiency. The goal is, like for the network and hardware parts, to make the comparison as fair as possible. Since LwM2M and MQTT represent respectively messaging and device management protocols, they do not share the same domain and their adaptions cannot therefore be identical. Starting from the security aspects, Leshan and Mosquitto use different DTLS and TLS libraries, respectively Scandium and OpenSSL. The first one is a Java and the second a C project. They have different scopes, the first adapted for multi-core machines and the latter for general usage, but come with the most appropriate open-source projects available for this thesis, as explained in Chapter 4.8. Next, the measurement must deal with the same cipher suites according to the security level. LwM2M Leshan for example uses by default an ECDHE suite for PSK. Unfortunately, Mosquitto does not. After adapting the Leshan code to include more ciphers on server-side, the cipher flag -c was set to TLS_PSK_WITH_AES_128_CCM_8: Advanced Encryption Standard with 128bit key in Counter with CBC-MAC mode with 8-Octet Integrity Check Value (ICV). The same was done for the MQTT client and broker as described in Listing 3.2. Focusing on the certificate settings, the experiment includes a high-computational cipher suite to monitor the protocol behaviors in challenging conditions. The cipher is called TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 and include: Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) as the key exchange protocol, Elliptic Curve Digital Signature Algorithm (ECDSA) as the authentication method, Advanced Encryption Standard with 128bit key in Galois/Counter mode (AES 128 GCM) as the encryption and Secure Hash Algorithm 256 (SHA256) as the hash algorithm. Leshan uses it by default while the Mosquitto client had to be modified. Moreover, during the first packet capture-tests the DTLS handshake in LwM2M showed a suboptimal performance. Table 5.2 reports the packet flow at server-side where the security handshake and later the LwM2M registration (application data) take place. The first noticeable detail is that frame 4 and 5 are equal. The server has in fact a DTLS retransmission timeout of 1 s as specified in RFC6347 (DTLS 1.2). Repeating the test multiple times, it becomes clear how this causes more traffic-over-the-air as well additive overhead at client-side. As a result, the server's experiment configuration was adjusted following RFC7252 (CoAP) which relaxes the retransmission timeout to 2 s. This change provided a faster client-response of about 25 ms in average. The second modification is triggered by focusing on the previous timeout-

cause. Frame 6 illustrates the packetization of certificate, client key exchange, certificate verify, change cipher spec and connection id in one single payload. It may be beneficial in terms of traffic but for a limited device like the Raspberry Pi the encryption of such components might require extra computational efforts and delays. After adapting the client to enable LwM2M DTLS packet-fragmentation, the new captures did not actually show neither consistent differences nor negative effects. Nevertheless, the measurements were carried on with this change as it represents a good-practice configuration for constrained devices.

As a matter of comparison, it is important to define the type of message sent by MQTT and LwM2M. The first has a flexibility in terms of QoS (for more information refer to Chapter 2.2.4). In the second one may choose between CON and NON CoAP POST (Chapter 4.8.1b). The objective is to configure MQTT to have least-consuming and best-effort network characteristics which make it suitable for LPWAN. These prerequisites translate into QoS 0 or fire and forget approach. Reflecting on LwM2M, the doubt was between making the protocol as similar as possible as MQTT or distancing it by keeping the pure UDP's connectionless nature. CON- allow a peer's confirmation adding limited delay and traffic, while NON-Posts are less robust but more lightweight. In the end, the experiment chooses to include NON confirmable messages. This is because also LwM2M has to adapt to real-world LPWAN providing a best-effort approach in case of bad coverage and low-signal conditions.

No.	Time (s)	Source	Destination	Protocol	Length (bytes)	Info
1	0.000000000	Client	Server	DTLSv1.2	156	Client Hello
2	0.000631922	Server	Client	DTLSv1.2	102	Hello Verify Request
3	0.142652410	Client	Server	DTLSv1.2	188	Client Hello
4	0.147935692	Server	Client	DTLSv1.2	1024	Server Hello, Certificate, Server Key Exchange, Certificate Request, Server Hello Done
5	1.148352071	Server	Client	DTLSv1.2	1024	Server Hello, Certificate, Server Key Exchange, Certificate Request, Server Hello Done
6	1.734453752	Client	Server	DTLSv1.2	923	Certificate, Client Key Exchange, Certificate Verify, Change Cipher Spec, Connection ID
7	1.744291004	Server	Client	DTLSv1.2	117	Change Cipher Spec, Encrypted Handshake Message
8	2.142554136	Client	Server	DTLSv1.2	231	Connection ID
9	2.144253597	Server	Client	DTLSv1.2	105	Application Data

Table 5.2: Packet Capture of LwM2M DTLS Handshake with Certificates

Finally, the measurement also includes a performance comparison using variable payload sizes. In this case, the choice was between including exactly the same payload or also consider the LwM2M data-encoding. As mentioned in Chapter 3.1.3, since LwM2M is a device management protocol, OMA's specification comprehends different formats encoding, e.g. Type-Length-Value (TLV). The same feature does not exist in

MQTT. The thesis decides to include LwM2M's choice and selects for MQTT the regular payload while for LwM2M its TLV encoding, adding some minimal length. For an 800 bytes regular unencrypted payload for example, TLV just adds 4 bytes (header).

5.3 Experiment Flow

The objective of the experiment is to compare LwM2M and MQTT analyzing the network and energy consumptions behaviors (find exact details at the end of this section). Variety is important to provide the reader use-cases where to apply the thesis' outcome. The first variable is the LPWAN access technology:

- LTE-M
- NB-IoT

Being the two most popular licensed cIoT links, it is interesting to see which IoT protocol is more suitable to which technology. The second variable is the authentication or more in general security method. There are three options:

- no Security (noSec)
- Pre-Shared-Key (PSK)
- Certificates (Cert)

All three are available for both TLS and DTLS, the security protocols utilized by the chosen MQTT and LwM2M projects respectively. As written in Chapter 5.2.3, the protocols are configured to operate with the same cipher suites and standard version, TLS and DTLS 1.2. Although the no security mode is hardly used in real-world implementations, it helps recognizing the trade-off between performance and safety provided by the measurements. The third variable regards typical IoT device activities when interacting with a server. The thesis focuses on four phases:

- Initial Connection
- Single Device to Server Message with payload sizes of 100, 200, 400, 800, 1600 and 3200 bytes
- Steady-State Update
- Mobility

The Initial Connection consists in a DTLS/TLS handshake plus a protocol-layer registration: either LwM2M Register operation or MQTT Connect followed by the related responses. The single Device to Server Message corresponds to either a LwM2M notification, a NON CoAP Post following an observe request by the server, or to an MQTT publish with QoS 0. Please notice that for the scope of this experiment the MQTT publish only reaches the broker. For this phase the experiment considers 6 kind of messages according to the payload involved: 100, 200, 400, 800, 1600 and 3200 bytes. The Steady-State Update is on the other hand either a periodical LwM2M update operation or an MQTT PING request. Please consider the difference between the two actions. In the first one the client performs an abbreviate handshake with the server using CID, while in the second one the client communicates a keep-alive message. Although diverse, Update and PING are compared with each other to once again deliver a real-world use-case where a device signalizes its state. The Mobility experiment diverges from the previous phases. In fact, it is a one-shot test where the device is immersed in an urban context. Using private or public transportation the modem moves within a city either covered by LTE-M or NB-IoT. The result should give an initial answer about the protocols' performances in a dynamic scenario.

Thanks to the tools described in Chapter 4.7, the measurements of the various scenarios defined above consisted in the following analysis:

- Packet Capture:
  - Packet Loss
  - Bytes Exchanged
  - Packets Exchanged
- Energy Consumption

The Packet Captures take place at server or broker side. Only during the Mobility Test packets are to be also captured at client-side. The *Packet Loss* corresponds for LwM2M to the packets which did not reach the server while for MQTT to the ones which were retransmitted before reaching either client or server. The Loss does not include duplicate packets. The Packets Exchanged is the number of packets which leave and reach the server including retransmissions and duplicates. The *Byte Exchanged*, on the other hand, is the sum of the packets exchanged length for a specific capture. Captures are limited to the time-frame in which a power measurement is performed. With Energy Consumption, this work defines the average energy consumed by the extension board (see Chapter 4.4) as long as a single action takes place. An action corresponds to handshake and protocol connection for the Initial Connection, packet send and related confirmations (applicable just to MQTT) for Single Device to Server Message as well as 10- and 25-minutes measurements for Steady-State Update and Mobility. The measurements are repeated 10 times and the evaluations report on average values for each Initial Connection and Single Device to Server Message scenario.

5.4 Measurements Details

The measurements took place between 13th and 23rd April 2021. The first part focused on LTE-M. For this purpose, the Cat-M1 modem connected to a testing network operated by Orange a.s. in Bratislava, Slovakia. For the Initial Connection, Single Device to Server Message and Steady-State phases, the device always remained at the same location in the city in a static mode without any antenna-attenuator. The research averages packet captures and average energy consumption results, obtained by performing 10 measurements for each specific configuration. The next section defines these more in detail. The LTE-M mobility test took place on 16th April 2021 travelling by public transportation within the city-borders for a 50 minutes total time-frame. On the other hand, for the NB-IoT part the modem used the access provided by Vodafone Italia in Milan, Italy. The conditions were identical as the one described for LTE-M, with the only difference that the signal conditions were different (please refer to Chapter 5.2.1 for more information). This time the mobility measurements were conducted on 23rd April 2021 from a moving car travelling again within the city-borders. For both dynamic experiments MQTT and LwM2M with PSK are the only two considered scenarios.

Table 5.3 reports all of the tests performed and later analyzed in Chapter 6. These are completed for both LTE-M and NB-IoT. The scenarios are primarily divided according to the device activity or phase. For every of these the device executes LwM2M and MQTT clients in distinct moments with different levels of security: noSec, PSK and Cert. Please refer to Table 6.8 to comprehend how to set the flags in the LwM2M client in order to reproduce the experiment. In line with the corresponding scenario, there are then additional configurations like the DTLS or TLS Version and Cipher Suites. Data interval and payload size refer primarily to the Single Device to Server Message activity and secondarily to the Mobility measurements. Finally, the Update or PING interval is relevant

in both Steady-State Update and Mobility phases. As mentioned in the previous section, every specific case (excluding the Mobility and Steady-State Update which are one-shot for respectively 25 and 10 minutes) consist of 10 distinct measurements based on events' (data transmission) logs and current peaks. The average energy consumption analysis' as well as the packet capture's results are then averaged to give an overview about the outcome, which is presented in Chapter 6.

Phase	Protocol	Security	DTLS/ TLS Ver- sion	Cipher Suite	Data Interval (s)	Payload Size (bytes)	Update/PING Interval (s)
Initial Connec- tion	LwM2M	noSec	1.2	-	-	-	-
	MQTT						
	LwM2M	PSK		TLS_PSK_WITH_AES_128_CCM_8			
	MQTT						
	LwM2M	Cert		ECDHE-ECDSA-AES128-GCM-SHA256			
	MQTT						
Single Device to Server Mes- sage	LwM2M	noSec		-	15	100 200 400 800 1600 3200	-
	MQTT					100 200 400 800 1600 3200	
	LwM2M	PSK		TLS_PSK_WITH_AES_128_CCM_8		100 200 400 800 1600 3200	
	MQTT					100 200 400 800 1600 3200	

	LwM2M	Cert		ECDHE-ECDSA-AES128-GCM-SHA256		100 200 400 800 1600 3200	
	MQTT					100 200 400 800 1600 3200	
Steady-State Update	LwM2M	noSec		-	-	-	15
							60
	MQTT						15
							60
	LwM2M	PSK		TLS_PSK_WITH_AES_128_CCM_8			15
							60
	MQTT						15
							60
	LwM2M	Cert		ECDHE-ECDSA-AES128-GCM-SHA256			15
							60
	MQTT						15
							60
Mobility	LwM2M	PSK		TLS_PSK_WITH_AES_128_CCM_8	15	400	60
	MQTT						

Table 5.3: Experiments' scenarios and configuration

6 Results and Analysis

This Chapter provides the experimental results and their evaluation. The four sections match four IoT devices' activities: Initial Connection, Single Device to Server Message, Steady-State Update and Mobility.

6.1 Initial Connection

Figure 6.1 shows the average bytes transferred between Server/Broker and Client during Initial Connection. The variation in percentage refers to the left-neighbor bar. In both LTE-M and NB-IoT, increasing the security level increases the transmitted data (the horizontal axis alternates MQTT and LwM2M with their level of securities). This happens because adding security-complexity translates into more encrypted data as well as more exchanged information. Cipher suites, certificates and key exchange are for example some of the blocks needed by both client and server for authentication when using DTLS with certificates. With PSK in comparison, only the client must send the key exchange while certificates are not involved. For LTE-M, overall LwM2M transfers in average 82% less bytes than MQTT's equivalent while in NB-IoT the difference reduces to 61%. Being a connectionless protocol, UDP allows keeping a lower bytes average than in TCP during initial connection. The cause is visible in Figure 6.2 and Figure 6.3 where MQTT needs more packets than each LwM2M's security-equivalent. There are also several MQTT packet losses occurring, culminating with MQTT Cert under NB-IoT with 3,6 average packet losses. These represent another reason of traffic-over-the-air in MQTT: every packet whose acknowledgment does not arrive at destination on time is retransmitted.

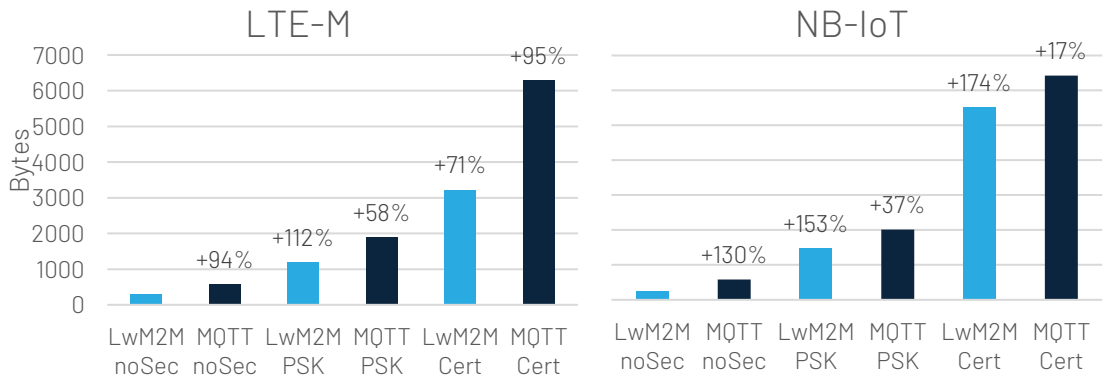

Figure 6.1: Average Bytes Exchanged during Initial Connection

Moreover, in Figure 6.1 there is an overall increase of bytes transferred in NB-IoT rather than in LTE-M which is mostly visible for LwM2M with Certificates (Cert) variating between 3219 and 5513 bytes. The same appears in Figure 6.2 and Figure 6.3 which also show a variation between the two access technologies: there is again a general growth in NB-IoT packets still mostly involving LwM2M Cert (from 13 in LTE-M to 21 in NB-IoT). This phenomenon has to do with the less stringent latency requirements in NB-IoT compared to LTE-M (Chapter 2.3). The transmission between peers takes longer and timeouts at both LwM2M client and server are often exceeded. The proof of this is in Figure 6.3, where LwM2M Cert in NB-IoT has a 7,7 average packet losses. As explained in Chapter 4.8, the experiment runs with an already adjusted timeout period of 2 seconds due to the late response probably caused by the DTLS' Java library that is not adapted

for constrained devices. This timeout period does not seem to be enough for NB-IoT. In fact, Table 6.1 shows the one packet capture for LwM2M Cert under NB-IoT. Excluding packets 1-5 which follow a different timeout configuration, packets 8-13 are sent and received twice. The reason is that packet 6 (server hello) did not receive an on-time response. The client certificate (packet 8) arrived with 177 ms delay indeed. Therefore, NB-IoT's extra latency combined with the constrained-devices unfriendly Java library causes higher transmission durations in NB-IoT's experiments, especially visible in LwM2M Cert. MQTT, on the other hand, utilizes OpenSSL, a TLS C library which shows better performances with certificates in constrained devices. This feature allows faster responses from the client. Nevertheless, as reported in Figure 6.3, 4 lost packets affect MQTT Cert, but only due to NB-IoT's high latency.

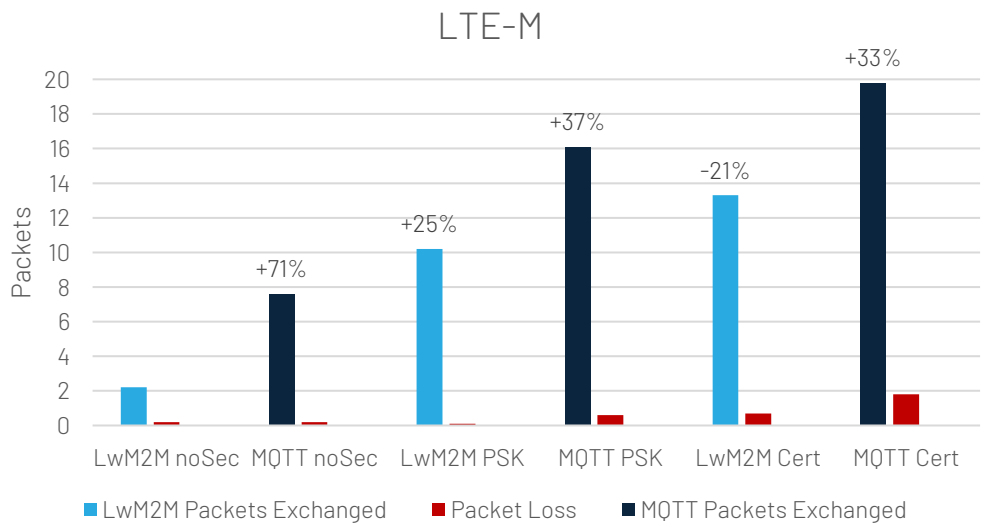

Figure 6.2: LTE-M - Average Packets Exchanged during Initial Connection

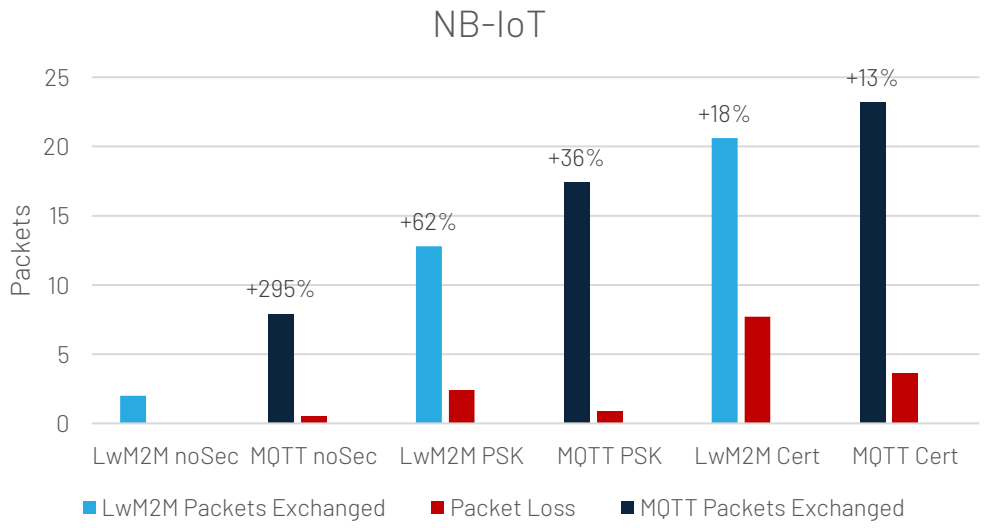

Figure 6.3: NB-IoT - Average Packets Exchanged during Initial Connection

No.	Time (s)	Source	Destination	Protocol	Length (bytes)	Info
1	0.000000000	Client	Server	DTLSv1.2	156	Client Hello
2	0.000419048	Server	Client	DTLSv1.2	102	Hello Verify Request
3	0.486722738	Client	Server	DTLSv1.2	156	Client Hello
4	0.487169565	Server	Client	DTLSv1.2	102	Hello Verify Request
5	0.599508005	Client	Server	DTLSv1.2	188	Client Hello
6	0.604988258	Server	Client	DTLSv1.2	1024	Server Hello, Certificate, Server Key Exchange, Certificate Request, Server Hello Done
7	2.605429974	Server	Client	DTLSv1.2	1024	Server Hello, Certificate, Server Key Exchange, Certificate Request, Server Hello Done
8	2.777431463	Client	Server	DTLSv1.2	677	Certificate
9	2.865512198	Client	Server	DTLSv1.2	133	Client Key Exchange
10	3.446252773	Client	Server	DTLSv1.2	141	Certificate Verify
11	3.446360076	Client	Server	DTLSv1.2	60	Change Cipher Spec
12	3.594256505	Client	Server	DTLSv1.2	110	Connection ID
13	3.594982537	Server	Client	DTLSv1.2	117	Change Cipher Spec, Encrypted Handshake Message
14	4.068979341	Client	Server	DTLSv1.2	677	Certificate
15	4.157390521	Client	Server	DTLSv1.2	133	Client Key Exchange
16	4.389119874	Client	Server	DTLSv1.2	141	Certificate Verify
17	4.389195438	Client	Server	DTLSv1.2	60	Change Cipher Spec
18	4.536901658	Client	Server	DTLSv1.2	110	Connection ID
19	4.538086667	Server	Client	DTLSv1.2	117	Change Cipher Spec, Encrypted Handshake Message
20	4.762286032	Client	Server	DTLSv1.2	231	Connection ID
21	4.763692673	Server	Client	DTLSv1.2	105	Application Data

Table 6.1: Packet Capture of LwM2M Cert in NB-IoT

By reporting the average energy consumptions through box-plots, Figure 6.4 shows a relevant increase in energy consumption within NB-IoT. For MQTT noSec for example in LTE-M the mean is 76.31 µWh while in NB-IoT 214.00 µWh. In order to investigate this more detail, Figure 6.6 illustrates energy consumption against the transmission duration using a scatter plot of all measurement samples, in both cellular technologies. LTE-M operates with compact Initial Connections durations approximately between 300 and 3200 ms while NB-IoT has a higher and broader extension between 700 and 9600 ms. This enlightens not only its worse latency performances but also its unpredictability: NB-IoT's sample points are more distant from each other than LTE-M's. Moreover, the transmission duration is directly proportional to the average energy consumption. A higher duration causes a greater consumption. This explains the results in Figure 6.4. Additionally, LTE-M's and NB-IoT's experiments took place in two different locations. This might have also influenced the performances. Nevertheless, the analysis of the signal conditions is out of this thesis' scope and was not further investigated.

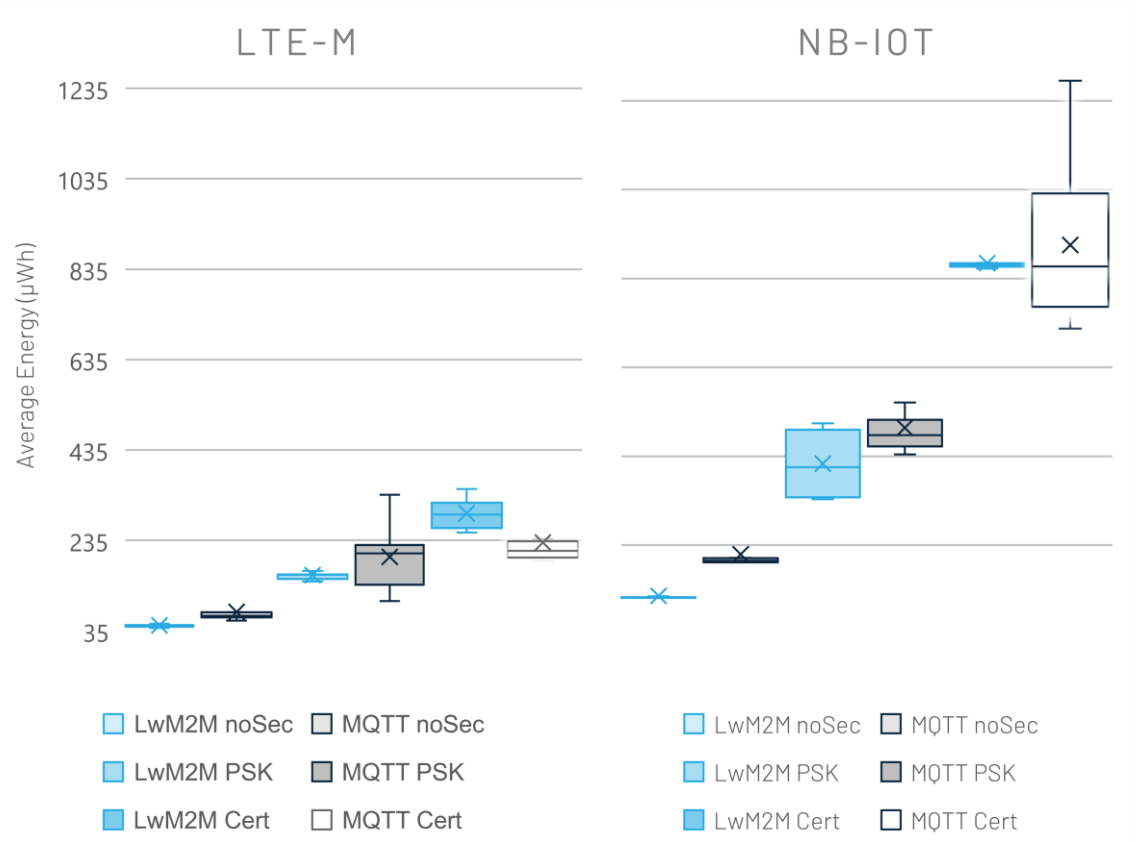

Figure 6.4: Average Energy Consumed during Initial Connection

Finally, once again the LTE-M energy consumptions box-plots in Figure 6.4 depict a higher average energy consumptions for LwM2M Cert rather than MQTT Cert. This is again due to LwM2M Cert the higher average initial connection duration in LTE-M as shown in Figure 6.5. The explanation relates again to Leshan's DTLS library, Californium/Scandium. It is designed as a server-side Java solutions for machines with multiple CPU cores that help handling many requests simultaneously [37]. During the experiment, the captures show a consistent delay by the client when delivering packets during the DTLS handshake. Nevertheless, because of the latency requirements in NB-IoT, MQTT Cert performs worse in terms of duration than LwM2M Cert, as illustrated in Figure 6.5. MQTT Cert variates its duration from 1600 in LTE-M to almost 6000 ms in NB-IoT. The causes here are not only the higher quantity of packets and bytes involved compared to LwM2M Cert, but also the multiple average packet losses and timeout retransmissions as described previously in Figure 6.2 and Figure 6.3.

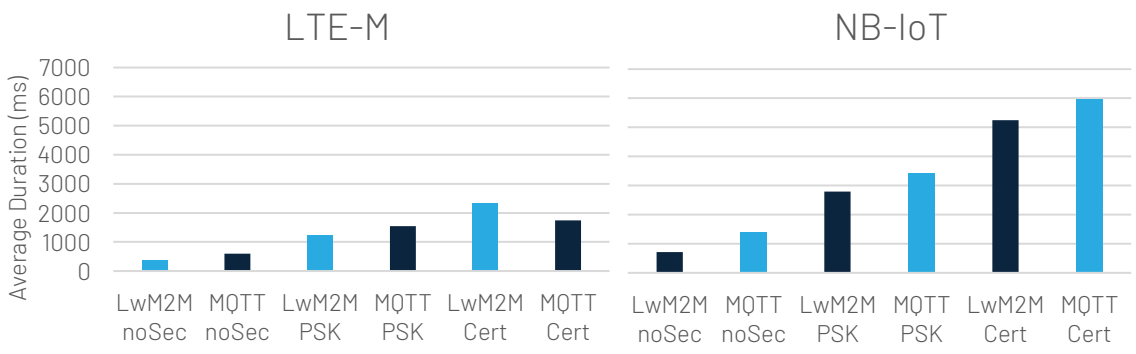

Figure 6.5: Average Duration during Initial Connection

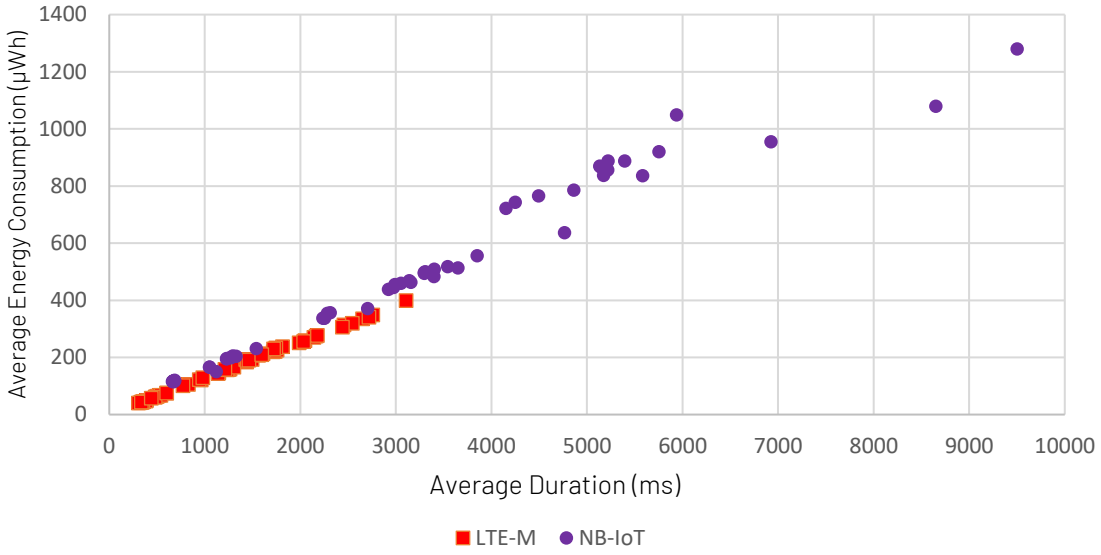

Figure 6.6: Relationship Duration vs Average Energy Consumption in LTE-M and NB-IoT

6.2 Single Device to Sever Message

Figure 6.7 and Figure 6.8 illustrate the average energy consumptions occurring when the modem transmits a single message with given payload 10 times in regular intervals of 15 seconds. The energy values refer to the time-frame when the device sends and receives all of the data necessary to complete the transfer. In LwM2M, they consist either in a single CoAP NON message if the payload does not exceed CoAP's Maximum Transmission Unit (MTU) of 1024 bytes or in multiple packets. This last case is called Block-Wise Transfer. On the other hand, MQTT operates in QoS 0 dealing with a minimum of 3 packets (three-way-handshake).

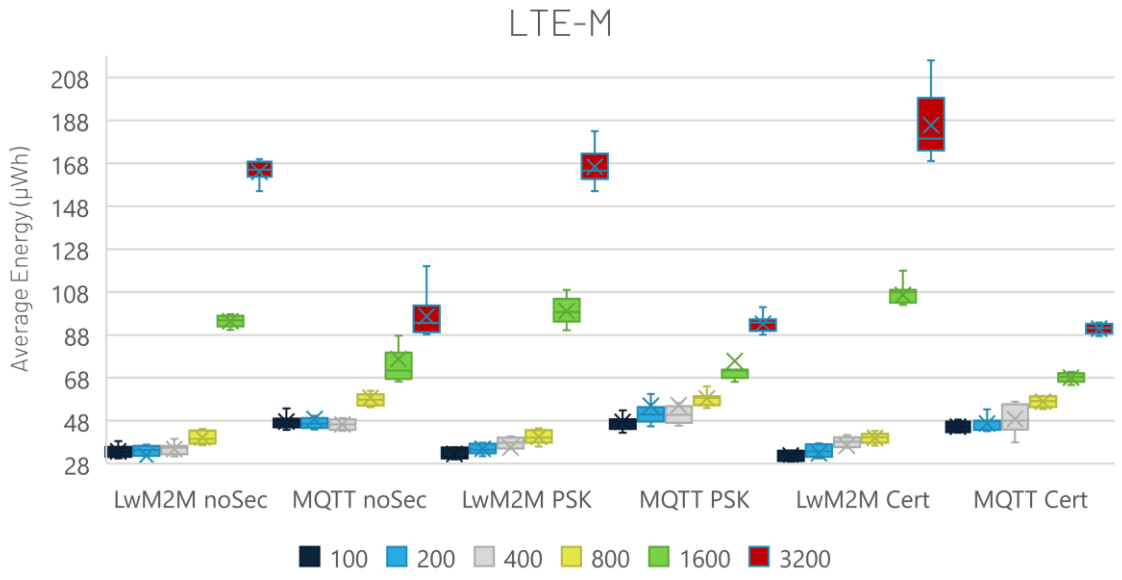

Figure 6.7: LTE-M - Average Energy Consumption during Single Device to Server Message with variable payload

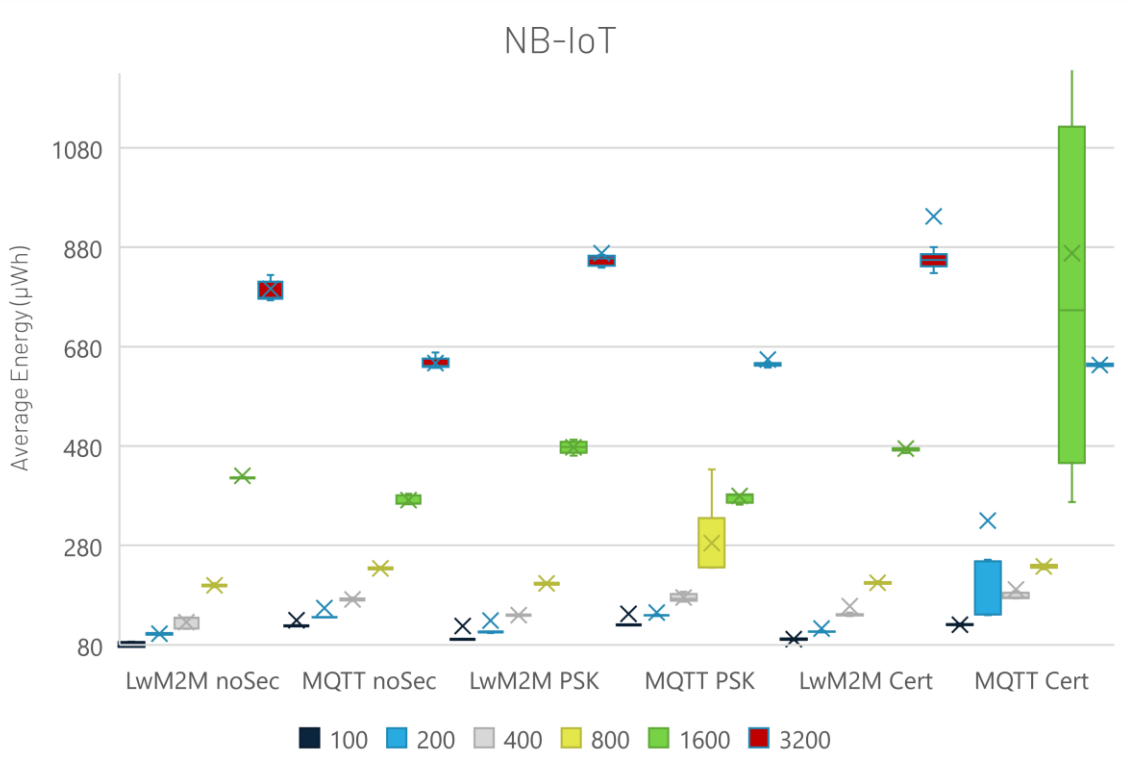

Figure 6.8: NB-IoT - Average Energy Consumption during Single Device to Server Message with variable payload

From the graphs, there are two clear trends. First, higher payloads correspond to higher energy consumptions. Secondly, with payloads between 100 and 800 bytes the LwM2M client consumes less average energy than MQTT with an equivalent level of security. For example, with 800 bytes payload, MQTT PSK uses +45 % average energy than LwM2M PSK. Contrarily with payloads higher than 800 bytes, MQTT outperforms LwM2M; for 3200 bytes payload, MQTT Cert consumes -51 % average energy than LwM2M Cert. The reason for this may become clear analyzing Figure 6.9 and Figure 6.10: the average number of packets involved in LwM2M for payloads equal to 1600 and 3200 bytes to send one single message is respectively 7 and 13 for both LTE-M and NB-IoT. The reason is the CoAP Block-Wise operation. The original payload is divided into multiple parts according to the library used. In this case, Leshan divides a 1600 bytes payload in one CoAP NON and three CON messages. These latter require a confirmation by the server and cause therefore an extra 3 packets being delivered to the client. For 3200 bytes payload there is 1 NON and 6 CON messages. This observation confirms the conclusions in the paper “Performance Evaluation of M2M Protocols Over Cellular Networks in a Lab Environment”:

> *"Especially in the case of LTE the transmission time depends not only on the total amount of data, but also on the exact sequencing of data-transfer. This has been clearly observed in the evaluation of CoAP. Its implementation of reliable data exchange is not suitable for the transmission of large payloads over cellular networks. Protocols based on TCP achieve a better performance due to TCP-features like windowing." [38, p. 6]*

An interesting result regards NB-IoT MQTT Cert - 1600 bytes transmission, which has been affected for several repetitions by a cellular link interruption. The proof of it, is the presence in Figure 6.9 and Figure 6.10 of a relatively high packet loss rate: for a total of

54 packets, 7 of them got lost. This introduced retransmissions, delays, and therefore higher average energy consumptions.

Once again, the average energy consumptions in NB-IoT reach higher values than in LTE-M. Despite the already mentioned reasons for the Initial Connection Experiment (higher latency in NB-IoT, higher duration as well as different signal conditions for LTE-M and NB-IoT), another singularity took place. The current consumption for the Single Device to Server Message experiment reached values above 350 mA for data-transfer. According to the official modem's data-sheet, the Quectel BG96 consumes a typical current of 157 mA operating in NB-IoT band 20 [33]. Also, adding 25 mA for LEDs and regulation losses, would not explain the peaks above 300 mA. It is not clear if the signal conditions itself may have caused the unexpected performance or if the reasons lies in other aspects.

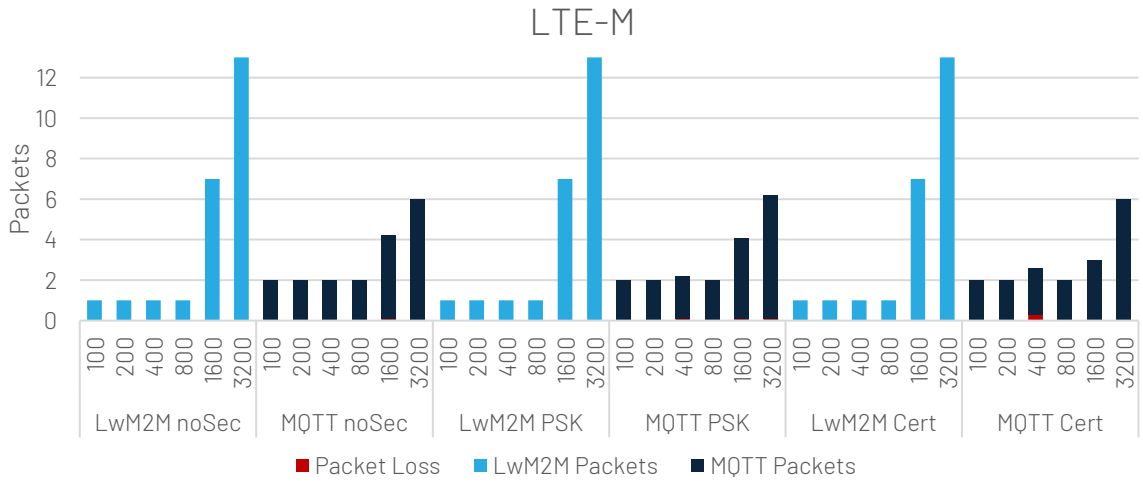

Figure 6.9: LTE-M - Average Packets Transmission during Single Device to Server Message with variable payload

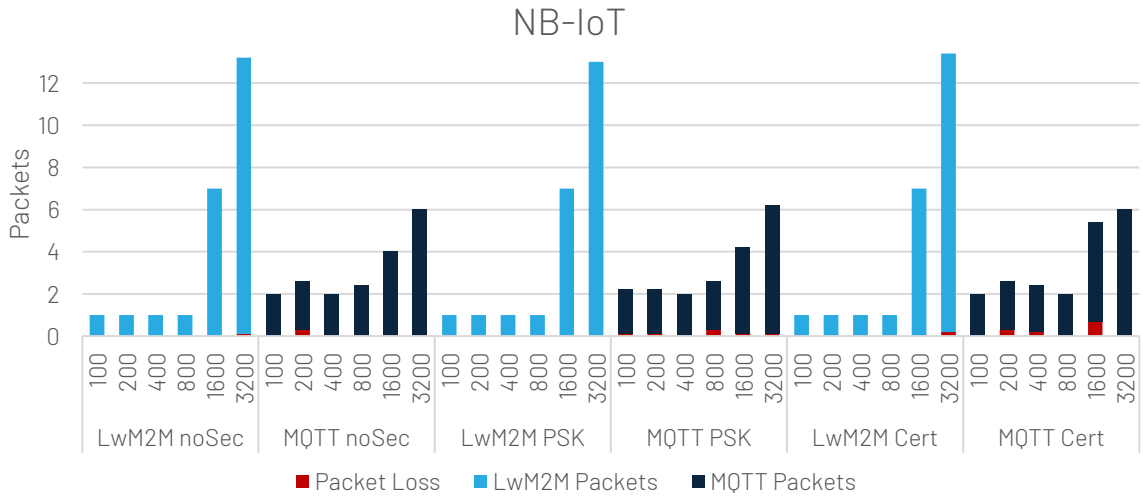

Figure 6.10: NB-IoT - Average Packets Transmission during Single Device to Server Message with variable payload

Finally, Figure 6.11 shows the average bytes within LTE-M and NB-IoT. It depicts a very similar behavior in both access technologies. This is because the notification operation in LwM2M (single data-transfer triggered by a server-observation) and the publish in MQTT are affected by much more relaxed time-outs periods than for the Initial Connection and handshake phases. The outcome is the almost absence of retransmissions (except for MQTT Cert – 1600 bytes) and therefore of added traffic-over-the-air.

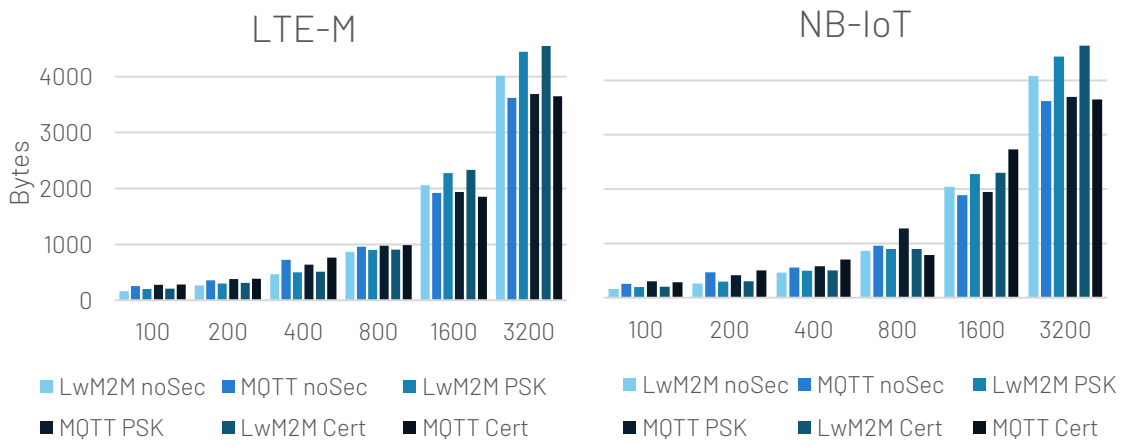

Figure 6.11: Average Bytes Transmission during Single Device to Server Message with variable payload

6.3 Steady-State Update

Whenever the device stops transmitting data it enters a so-called steady-state phase. In order to keep the actual connection to the server, the client needs to signalize its status by sending periodical messages. In LwM2M, the client performs the Update operation while in MQTT the most comparable action is the PING. Although they have different purposes, because there are no more similar equivalent operations between the protocols, the experiment Steady-State Update experiment chooses to compare Update and PING. For more information please refer Chapter 5.3. The two update intervals selected for the 10 minutes measurements were 15 and 60 seconds.

Figure 6.12 and Figure 6.13 illustrate how the number of bytes exchanged varies according to protocol and security method used. LwM2M always outperforms MQTT in terms of traffic-over-the-air. In some cases, MQTT uses more than 40 % more bytes than LwM2M's security-equivalent (the percentages in the graphs corresponds in each protocol/security case to the variation with their left-neighbor). There is also an increase between different security-levels within the same protocol, i.e. in LTE-M's 15 seconds update interval, LwM2M Cert involves 8120 bytes while LwM2M PSK just 7480. Thus, the encryption complexity influences the packet length. This holds true for any case but LTE-M's 60 seconds update interval where MQTT Cert just needs 2600 bytes while MQTT PSK 2671. Another important observation regards DTLS. Even though LwM2M's Update operation in the PSK and Cert cases represents a re-handshake (please find more information in Chapters 2.1.9 and 5.2.3) which may add complexity, CID minimizes it by obsoleting many DTLS handshakes. The positive effects are visible from Figure 6.12 and Figure 6.13: although it does not carry any handshake purposes, MQTT's PING still requires greater exchanges of bytes than LwM2M. The reason is its three-way communication. Please note that MQTT does not need a re-handshake like in DTLS, since TCP/TLS based protocols are connection-oriented and perform therefore return-routability checks as part of any connection establishment [39, p. 22].

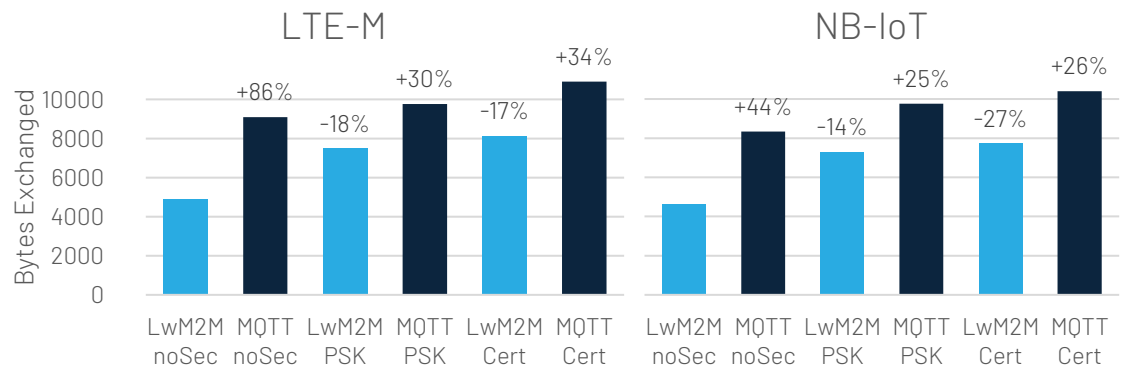

Figure 6.12: Average Bytes Exchanged for a 15 seconds Update Interval

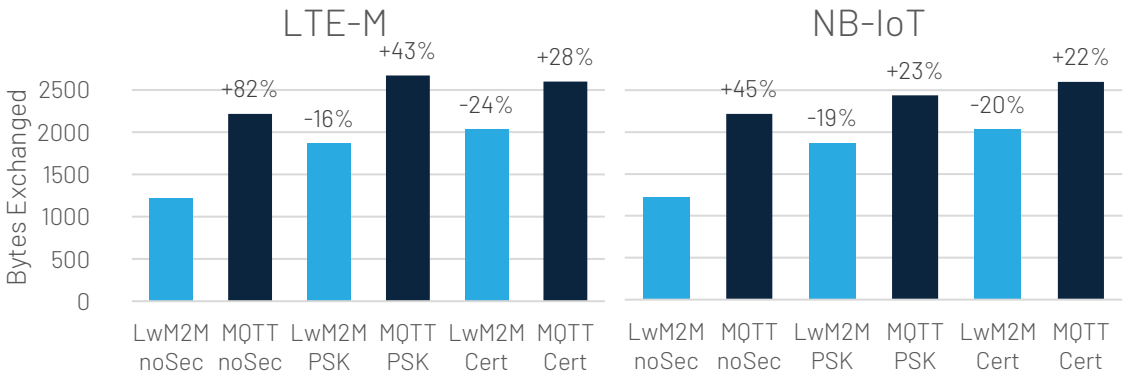

Figure 6.13: Average Bytes Exchanged for a 60 seconds Update Interval

The energy consumption analysis in Figure 6.14 for the 15 seconds update interval depicts once again the better performances achieved by LwM2M. As for the Initial Connection in Chapter 6.1 the network signal quality and NB-IoT's larger latencies also influenced this experiment. Nevertheless, one may notice lower average energy results in NB-IoT than in LTE-M. These are mainly caused by the diverse DRX cycle length: 2.56 s for LTE-M and 10.24 s as described in Table 5.1. A shorter DRX cycle means more paging windows and therefore higher average current and energy consumption.

Because the measurements took place for a 10 minutes time-frame, all energy consumptions detections for the Steady-State Update were affected by the 5 s RRC inactivity timer in both LTE-M and NB-IoT networks. This has an enormous effect on the registered average energy consumptions. For each effective update-data-exchange which lasts between 300 and 400 ms, there is a 5 s RRC inactivity timer window. Therefore, the real packet-transfer part only accounts for 5.7-7.4 % of the connected state for each update operation. This greatly reduces the impact of the used IoT protocols (LwM2M and MQTT) on the energy consumption. This is especially noticeable from Figure 6.15: the results' differences are sometimes too trifling to be validated. Therefore, the differences between the protocols performances become irrelevant for a long-term measurement. Various papers confirm the effect of the RRC inactivity timer. The authors of "Guidelines for an Energy Efficient Tuning of the NB-IoT Stack" [40] for example state the following: *"the tuning of the RRC inactivity timer has a significant impact on the device energy consumption"* [40, p. 9]. As explained in Chapter 2.3.9 a valid solution to this issue is the Release Assistant Indication (RAI). This is however only available in a limited number of modems and only within NB-IoT.

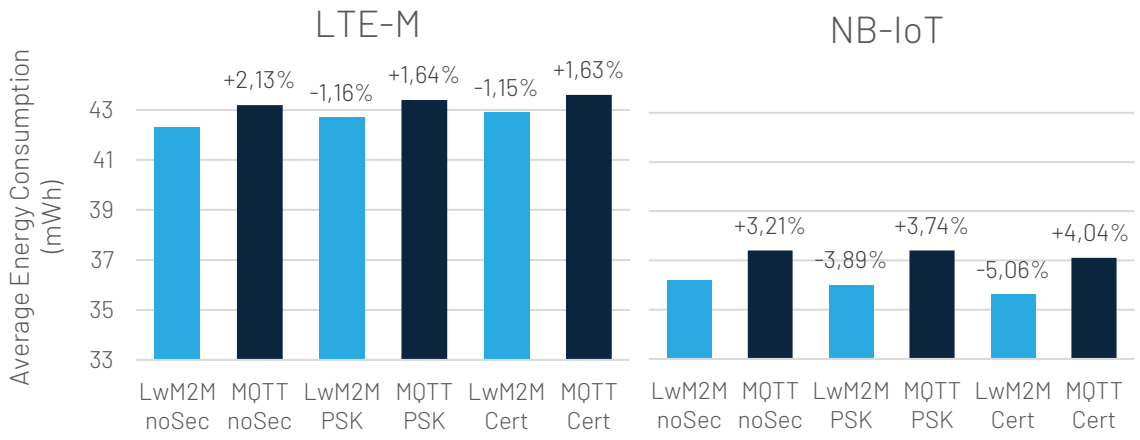

Figure 6.14: Average Energy Consumption for a 15 seconds Update Interval

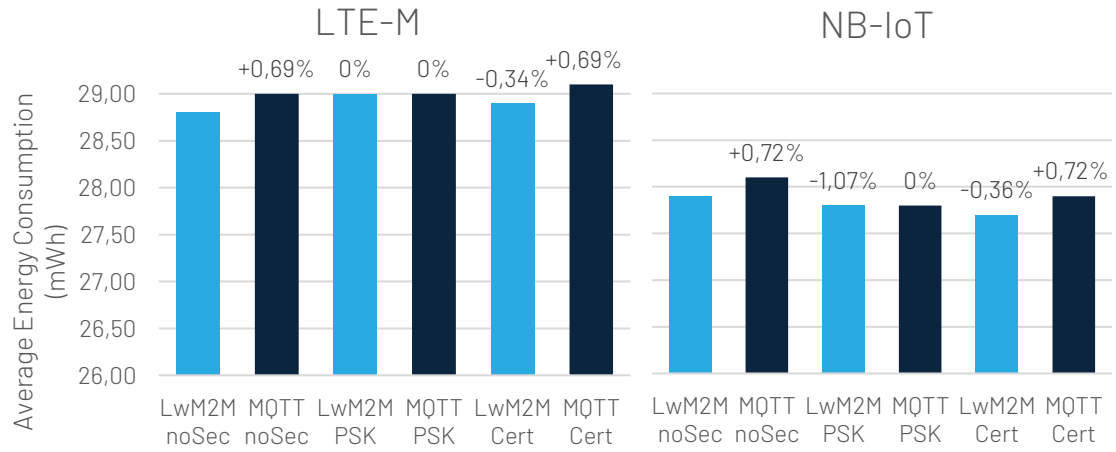

Figure 6.15: Average Energy Consumption for a 60 seconds Update Interval

6.4 Mobility

Because IoT concepts often involve dynamic scenarios like tracing, the Mobility experiment wants to give an example on how LwM2M and MQTT may perform when handovers and signal quality variations take place. Several not-controllable real-world aspects influenced the measurements: car traffic, transportation's speed, operator's coverage, and urban routes. The reader should therefore take in account the hard reproducibility of the presented results (for more details please refer to Chapters 5.3 and 5.4). Like in the previous section, the RRC inactivity timer and the absence of RAI make the average energy comparison less reliable.

Table 6.2 reports the LTE-M results measured in a 25 minutes time-frame moving by public transportation within the city of Bratislava, Slovakia. Both LwM2M and MQTT clients successfully deliver the totality of the notifications- and publish-messages. Nevertheless, MQTT also include 7 retransmissions. The number of packets sent by both clients differs due to the following factors: different handshake duration, coverage, and Observation handling for LwM2M. The notification is in fact triggered by the Observe operation requested by the Server, while MQTT may start publishing as soon as a session begins. LwM2M utilizes 22.8% less bytes and 51.37% less packets to accomplish initial connection, termination, notifications, and regular updates. Duplicates (not reported in the table) and retransmissions partly account for the higher byte and packet flow in

MQTT. The average energy consumption gives an idea on how the data transmissions impact on devices' battery-life. Due to the higher traffic generated, MQTT has 3.77% greater average energy than LwM2M while using LTE-M.

LTE-M	Var % in respect to MQTT PSK	LwM2M PSK	MQTT PSK	Var % in respect to LwM2M PSK
Bytes Exchanged	-22.80%	55086	71356	+29.54%
Packets Exchanged	-51.37%	160	329	+105.63%
Data Packets Sent by Client	-2.00%	98	100	+2.04%
Data Packets Received by Server	-2.00%	98	100	+2.04%
Packets Retransmissions		-	7	
Average Energy Consumption (mWh)	-3.64%	106	110	+3.77%

Table 6.2: LTE-M - LwM2M vs MQTT with PSK in the Mobility Experiment

During the second part of the measurements, which took place in Milan, Italy, while moving by car, the MQTT client suffered from a lack of NB-IoT coverage for a period of about 4 minutes. This caused on one hand high retransmission rates and packet losses. On the other, less traffic-over-the-air as less packets and bytes reached the server. Table 6.3 reports for MQTT in NB-IoT less bytes and packets exchanged than in LTE-M, respectively 69535 and 326 against 71356 and 329. Without coverage-losses these results would have been unexpected due to the longer latencies as experienced for NB-IoT in the experiment Initial Connection. The 4 minutes no-coverage-window actually did not prevent all of the packets to leave the client. In fact, only the data (13 packets) out of the keep-alive window of 60 seconds was never sent; the other packets reached the server once the connection was re-established. In a similar scenario, UDP-based protocols would have ignored all the losses but avoiding at the same time retransmissions.

As for the LTE-M case, LwM2M is more efficient in terms of bytes, packets, and average energy consumption. Nevertheless, there are more packets and bytes involved than in LTE-M. The reason are the stringent time-outs and therefore packets repetitions during Initial Connection, Termination and Updates. This is due to the higher latency in NB-IoT. As mentioned in Chapter 5.2.1, the BG96 adapts to the default DRX cycle period, which is longer than in LTE-M's Slovak network. The end results do not reflect this difference. There are lower energy consumptions in LTE-M. The reason is the following: the multiple transmission events (publish/notification and periodical update) followed by the 5 s RRC inactivation timer, left few idle mode windows where the DRX cycle might take place. The consequence is a low or absent influence of the DRX cycle length. As already mentioned, the number of retransmissions due to time-outs are the most influencing factors in terms of energy consumption in this case.

Although MQTT was affected by a coverage-loss, the Mobility experiment in NB-IoT shows LwM2M's better adaptability to network delays caused by higher latencies. Another reason for this is the concept explained in Chapter 3.2.4. LTE-M handles cell-handovers in both idle and connected modes, while NB-IoT just in idle. This causes higher packet losses as well as larger battery consumptions due to retransmissions when using MQTT. This protocol in fact operates better in LTE-M due to the higher connection regularity.

NB-IoT	Var % in respect to MQTT PSK	LwM2M PSK	MQTT PSK	Var % in respect to LwM2M PSK
Bytes Exchanged	-16,89%	57789	69535	+20,33%
Packets Exchanged	-37,42%	204	326	+59,80%
Data Packets Sent by Client	-4,00%	96	100	+4,17%
Data Packets Received by Server	+8,05%	94	87	-7,45%
Packets Retransmissions		-	56	
Average Energy Consumption (mWh)	-4,84%	118	124	+5,08%

Table 6.3: NB-IoT - LwM2M vs MQTT with PSK in the Mobility Experiment

Summary and Outlook

The scope of this work is to support academics, enterprises, and end-users to tackle the following IoT challenges: limited battery-life, growing traffic-over-the-air and remote connectivity. Therefore, this master's thesis investigates and compares the performances of LwM2M and MQTT operating in LTE-M and NB-IoT. First, the theory provides the details to understand the topics involved. Then, an explanation regarding hardware, software and network used in the research follows. Subsequently, a preliminary test of the tools allows to understand and adjust configurations before the measurements take place. In the end, comparison and evaluation are supported by a network analysis involving packet and bytes exchanged as well as measurements delivering the average energy consumption of an IoT constrained device. The experiments include a client-server communication via LTE-M and NB-IoT during four typical IoT transmission's events: initial connection, single client to server message, steady-state update, and mobility scenario. Moreover, the study analyzes these phases applying three different client-server authentication methods: no-security, PSK and certificates.

The work shows that LwM2M transfers in average 82% and 61 % less bytes than MQTT respectively in LTE-M and NB-IoT during initial connection. About this matter, it turns out that choosing less stringent timeouts configurations as well as appropriate TLS/DTLS libraries for constrained devices, helps reducing energy consumptions and traffic-over-the-air. This is especially true for NB-IoT where latencies are higher than LTE-M. Moreover, when sending single messages from client to server, MQTT outperforms LwM2M for large payload-transfer, consuming 20% less average energy. On the other hand, when maintaining packet lengths below LwM2M's MTU, MQTT uses 44% more average energy. In general, the measurements also prove that enhancing the security level complexity, data-usage and energy consumptions also increase. The experiments also dealt with 10- and 25-minutes time-windows measurements during steady-state update and mobility scenarios. In both cases, the cellular networks' non-tunable RRC inactivation timer made the IoT protocols' impact, on the device's energy consumption, negligible. Nevertheless, choosing a longer DRX cycle helps reducing energy consumptions as shown in NB-IoT for the steady-state update experiment.

The thesis did not focus on acquiring signal quality parameters like RSSI, RSRP and RSRQ along with the energy consumption and packet analysis. This would be relevant to find a relationship between signal conditions, latency, energy consumptions and packet behaviors. Also, the IoT cellular coverage analysis was out of scope. Another open aspect is the evaluation of LwM2M and MQTT in devices enabling Power Saving Mode (PSM) within LTE-M and NB-IoT. A measurement lasting hours, days or even months would be interesting to understand the protocols' performances exploiting additional features of cIoT. Moreover, as previously mentioned, there is a non-adjustable network parameter: the RRC inactivity timer. This highly influences the devices' energy consumptions and is controlled by cellular operators. Therefore, a future research involving the Release Assistant Indicator (RAI) would be beneficial to understand the real performances of LwM2M and MQTT on longer time-intervals. Finally, version 1.2 of the OMA specification includes LwM2M working on top of MQTT. As of May 2021, no open-source repository contains this feature. Once this will become available, a further investigation may result beneficial for enterprises and users who are considering adopting LwM2M over MQTT.

Appendix

A. LTE-M and NB-IoT Operators

Country	Operator	PLMN ID	Technology	Deployment Bands
Germany	Deutsche Telekom	26201	LTE-M	20, 3
Netherlands	T-Mobile Netherlands	20416	LTE-M	8
Netherlands	KPN	20408	LTE-M	20
Netherlands	Vodafone Libertel	20404	LTE-M	20
Austria	Magenta Telekom	23203	LTE-M	20, 3
France	Orange	20801	LTE-M	20
Belgium	Orange	20610	LTE-M	20
Slovakia	Orange	23101	LTE-M	20
Switzerland	Swisscom	22801	LTE-M	20
Latvia	LMT	24701	LTE-M	20
Germany	Deutsche Telekom	26201	NB-IoT	8
Germany	Vodafone	26202	NB-IoT	20
Netherlands	T-Mobile Netherlands	20416	NB-IoT	8
Netherlands	Vodafone Libertel	20404	NB-IoT	20
Austria	Magenta Telekom	23203	NB-IoT	8
Czech Republic	T-Mobile Czech	23001	NB-IoT	20
Slovakia	Slovak Telekom	23102	NB-IoT	20
Poland	T-Mobile Poland	26002	NB-IoT	20
Croatia	Hrvatski Telekom	21901	NB-IoT	20
Hungary	Magyar Telekom	21630	NB-IoT	20, 8
Greece	Cosmote	20201	NB-IoT	20
Switzerland	Swisscom	22801	NB-IoT	20
Liechtenstein	Swisscom	22801	NB-IoT	20
Italy	Vodafone	22210	NB-IoT	20
Italy	TIM	22201	NB-IoT	20
Spain	Vodafone	21401	NB-IoT	20
United Kingdom	Vodafone	23415	NB-IoT	20
Sweden	Telia	24001	NB-IoT	20
Finland	Telia	24491	NB-IoT	20, 3
Denmark	Telia	23820	NB-IoT	20, 8
Norway	Telia	24202	NB-IoT	20

Belgium	Telenet	20620	NB-IoT	20
Belgium	Orange	20610	NB-IoT	20
USA	AT&T	310410	LTE-M	2, 4, 12
USA	T-Mobile US	310260	NB-IoT	2, 4, 12, 66
Japan	NTT DoCoMo	44010	LTE-M	1, 19

Table 6.4: 3GPP's CIoT Operators Details [28]

B. AT Commands procedure for NB-IoT

```
// NB-IoT scan mode configuration
AT+QCFG="nwscanseq",03,1
AT+QCFG="nwscanmode",3,1
AT+QCFG="iotopmode",0,1

// Set NB-IoT Bands to LTE Band 20
AT+QCFG="band",0,80000,0

// Define PDP context using 1NCE APN
AT+CGDCONT=1,"IP","iot.1nce.net",,

// Manual Registration to Vodafone Italia
AT+COPS=1,2,"22210",9

// Retrieve network and operator info
AT+QNWINFO
```

Listing 6.1: NB-IoT Modem connection procedure through AT Commands

C. Sixfab Cellular IoT HAT

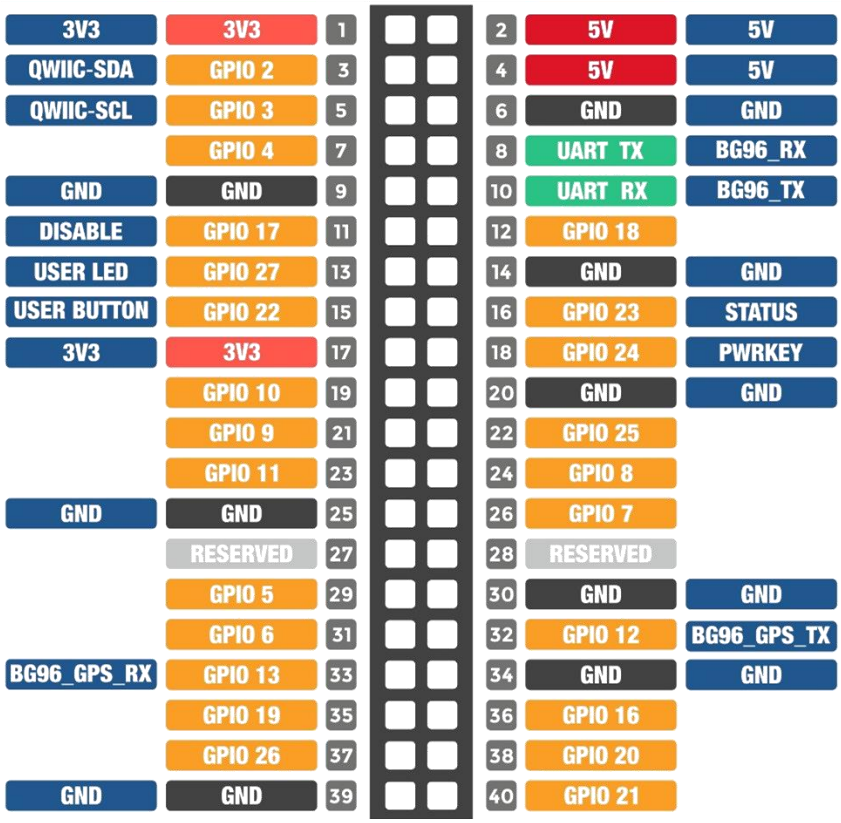

Figure 6.16: Sixfab Cellular IoT HAT Pinout Schematics [33]

Description	Conditions	Typ.	Unit
OFF State	Power down	8	µA
PSM	Power Saving Mode @Real Network	10	µA
Quiescent Current	AT+CFUN=0 @Sleep State	0.8 *	
Sleep State	DRX=1.28s @Real LTE Cat M1 Network DRX=1.28s @Real LTE Cat NB1 Network e-I-DRX=20.48s @Real LTE Cat M1 Network e-I-DRX=20.48s @Real LTE Cat NB1 Network @Real 2G Network	1.7 * 2.2 1.1 1.7 2.0 *	mA
Idle State	DRX=1.28s @Real LTE Cat M1 Network DRX=1.28s @Real LTE Cat NB1 Network e-I-DRX=20.48s @Real LTE Cat M1 Network e-I-DRX=20.48s @Real LTE Cat NB1 Network @Real 2G Network	16 * * 16 * 15 * 15 * 15 * *	mA

Description	Conditions	Typ.	Unit
LTE Cat M1 data transfer (GNSS OFF)	LTE-FDD B1 @23.31dBm LTE-FDD B2 @23.05dBm LTE-FDD B3 @23.09dBm LTE-FDD B4 @23.19dBm LTE-FDD B5 @23.22dBm LTE-FDD B8 @21.83dBm LTE-FDD B12 @21.88dBm LTE-FDD B13 @21.96dBm LTE-FDD B18 @23.04dBm LTE-FDD B19 @23.13dBm LTE-FDD B20 @23.07dBm LTE-FDD B26 @22.81dBm LTE-FDD B28 @22.52dBm LTE-TDD B39 @TBD	220 208 214 214 210 203 215 197 212 211 209 214 215 TDB	mA
LTE Cat NB1 data transfer (GNSS OFF)	LTE-FDD B1 @22.8dBm LTE-FDD B2 @22.6dBm LTE-FDD B3 @22.6dBm LTE-FDD B4 @22.6dBm LTE-FDD B5 @22.9dBm LTE-FDD B8 @22.7dBm LTE-FDD B12 @23dBm LTE-FDD B13 @22.9dBm LTE-FDD B18 @23.1dBm LTE-FDD B19 @22.9dBm LTE-FDD B20 @22.7dBm LTE-FDD B26 @22.8dBm LTE-FDD B28 @22.5dBm	170 171 161 161 156 170 170 167 159 155 157 162 163	mA

Table 6.5: Sixfab Cellular IoT HAT Technical Details

Notes:

*Typical value with USB and UART disconnected.

**Sleep state with UART connected and USB disconnected. The module can enter into a sleep state
through executing AT+QSCLK=1 command via UART interface and then controlling the module’s
DTR pi

***Idle state with UART connected and USB disconnected.

These values refer to the consumption of only the BG96 module, not the whole circuit at all. With the LEDs and regulation losses, it may rise to 25mA.

D. Qoitech Otii Arc

Type	Min	Unit	Max
Operating environment	15 °C / 60 °F		25 °C / 77 °F
USB Output voltage (auto range)	0.5 V		3.75 V
USB Output voltage (locked to high current range)	0.5 V		4.2 V
USB Output voltage setting resolution		1 mV	
USB Output current		250 mA	
External Output voltage (auto range)	0.5 V		4.55 V
External Output voltage (locked to high current range)	0.5 V		5.0 V
External Output voltage setting resolution		1 mV	
External Output current, max continuous		2.5 A	
External Output current, max peak		5 A	
Accuracy Analog		± (0.1% + 50 nA)	
Sample Rate in ±19 mA range		4 ksps	
Sample Rate in ±2.7A range		1 ksps	
Sample Rate in ±5.0 A range		1 ksps	
bandwidth (3 dB)		400 Hz	
Total accuracy		± (0.1% + 1.5 mV)	
Sample Rate		1 ksps	
Bitrate	110 bps		5.25 Mbp

Table 6.6: Otii Arc Specs

E. LwM2M Leshan’s Server and Client Flags

Flag	Explanation
-lh,--coaphost <arg>	Set the local CoAP address
-lp,--coapport <arg>	Set the local CoAP port
-slh,--coapshost <arg>	Set the secure local CoAP address
-slp,--coapsport <arg>	Set the secure local CoAP port
-wh,--webhost <arg>	Set the HTTP address for web server
-wp,--webport <arg>	Set the HTTP port for web server
-m,--modelsfolder <arg>	A folder which contains object models in OMA DDF(.xml) format
-oc	Activate support of old/deprecated cipher suites
-cid <arg>	Control usage of DTLS connection ID: - 'on' to activate Connection ID support (same as -cid 6) - 'off' to deactivate it - Positive value define the size in byte of CID generated. - 0 value means we accept to use CID but will not generated one for foreign peer
-r,--redis <arg>	Use redis to store registration and securityInfo
-mdns,--pub-lishDNSSdServices	Publish leshan's services to DNS Service discovery
-pubk <arg>	The path to your server public key file
-prik <arg>	The path to your server private key file
-cert <arg>	The path to your server certificate or certificate chain file
-truststore <arg>	The path to a root certificate file to trust or a folder containing all the trusted certificates in X509v3 format (DER encoding) or trust store URI
-ks,--keystore <arg>	Set the key store file
-ksp,--storepass <arg>	Set the key store password
-kst,--storetype <arg>	Set the key store type
-ksa,--alias <arg>	Set the key store alias to use for server credentials
-ksap,--keypass <arg>	Set the key store alias password to use

Table 6.7: Flags for executing the Leshan Demo Server

Flag	Explanation
-n <arg>	Set the endpoint name of the Client
-b	If present use bootstrap
-l <arg>	The lifetime in seconds used to register, ignored if -b is used
-cp <arg>	The communication period in seconds which should be smaller than the lifetime, will be used even if -b is used
-lh <arg>	Set the local CoAP address of the Client
-lp <arg>	Set the local CoAP port of the Client
-u <arg>	Set the LWM2M or Bootstrap server URL
-r	Force reconnect/re-handshake on update
-f	Do not try to resume session always, do a full handshake
-ocf	activate support of old/unofficial content format
-c <arg>	Define cipher suites used
-oc	Activate support of old/deprecated cipher suites
-cid <arg>	Control usage of DTLS connection ID. - 'on' to activate Connection ID support (same as -cid 0) - 'off' to deactivate it - Positive value define the size in byte of CID generated. - 0 value means we accept to use CID but will not generated one for foreign peer
-aa <arg>	Use additional attributes at registration time
-bsaa <arg>	Use additional attributes at bootstrap time
-m <arg>	A folder which contains object models in OMA DDF(.xml)for-mat
-pos <arg>	Set the initial location (latitude, longitude) of the device to be reported by the Location object
-sf <arg>	Scale factor to apply when shifting position
-bs <arg> (added)	Set observation Byte Size
-i <arg>	Set the LWM2M or Bootstrap server PSK identity in ascii
-p <arg>	Set the LWM2M or Bootstrap server Pre-Shared-Key in hexa
-cpubk <arg>	The path to your client public key file
-cprik <arg>	The path to your client private key file
-spubk <arg>	The path to your server public key file
-ccert <arg>	The path to your client certificate file
-scert <arg>	The path to your server certificate file
-truststore <arg>	The path to a root certificate file to trust or a folder containing all the trusted certificates in X509v3 format (DER encoding) or trust store URI

-cu,--certificate-usage <arg>	Certificate Usage (as integer) defining how to use server certificate. - 0: CA constraint - 1: service certificate constraint - 2: trust anchor assertion - 3: domain issued certificate (Default value)

Table 6.8: Flags for executing the modified Leshan Client Demo

References

[1] Altexsoft, *Making Sense of IoT Platforms: AWS vs Azure vs Google vs IBM vs Cisco.* [Online]. Available: https://www.altexsoft.com/blog/iot-platforms/ (accessed: 19th May 2021).

[2] Steve, “LwM2M-and-MQTT-test-lab-comparison-MachNation-MIT-E-v1.3,”

[3] S.-C. Son, S. Ko, H. Lee, and B.-T. Lee, “LwM2M based IoT Microservice Model with Repicas Synchronization Technique,” in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju, Korea (South), 10212020, pp. 1802–1804.

[4] J. Wirges and U. Dettmar, “Performance of TCP and UDP over Narrowband Internet of Things (NB-IoT),” in *2019 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS)*, BALI, Indonesia, 112019, pp. 5–11.

[5] Software AG, *Software AG Technology Radar for IoT and Integration.* [Online]. Available: https://techradar.softwareag.com/ (accessed: Mar. 2 2021).

[6] OMA Specworks, “OMA-TS-LightweightM2M_Transport-V1_2-20201110-A_full,” OMA Specworks, Nov. 2020. Accessed: Feb. 19 2020.

[7] OMA Specworks, “OMA-TS-LightweightM2M_Core-V1_2-20201110-A_full: Approved Version: 1.2 - 2020-11-10,” 2020.

[8] P. Wouters, H. Tschofenig, J. Gilmore, S. Weiler, and T. Kivinen, “Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS),” 2014.

[9] OASIS Message Queuing Telemetry Transport (MQTT) TC, “MQTT Version 5.0,”

[10] G. Restuccia, H. Tschofenig, and E. Baccelli, “Low-Power IoT Communication Security: On the Performance of DTLS and TLS 1.3,” in *2020 9th IFIP International Conference on Performance Evaluation and Modeling in Wireless Networks (PEMWN)*, Berlin, Germany, Dec. 2020 - Dec. 2020, pp. 1–6.

[11] M. B. Yassein, M. Q. Shatnawi, S. Aljwarneh, and R. Al-Hatmi, “Internet of Things: Survey and open issues of MQTT protocol,” in *2017 International Conference on Engineering & MIS (ICEMIS)*, Monastir, 52017, pp. 1–6.

[12] Jens Deters, *Hello MQTT Version 5.0!* [Online]. Available: https://blog.codecentric.de/en/2017/11/hello-mqtt-version-5-0/ (accessed: Apr. 2 2021).

[13] J. Wirges, “Design Of A Narrowband Internet Of Things (NB-IoT) Low-Power Communication Module For Cloud-Based IoT Applications,” Master Thesis, TH Köln, University of Applied Sciences, Cologne, 2019. Accessed: Apr. 3 2021.

[14] J. Toldinas, B. Lozinskis, E. Baranauskas, and A. Dobrovolskis, “MQTT Quality of Service versus Energy Consumption,” in *2019 23rd International Conference Electronics*, Palanga, Lithuania, 62019, pp. 1–4.

[15] The HiveMQ Team, *Quality of Service 0,1 & 2 - MQTT Essentials: Part 6: MQTT Essentials MQTT* (accessed: Apr. 3 2021).

[16] S. Hernández Ramos, M. T. Villalba, and R. Lacuesta, “MQTT Security: A Novel Fuzzing Approach,” *Wireless Communications and Mobile Computing*, vol. 2018, pp. 1–11, 2018, doi: 10.1155/2018/8261746.

[17] O. Liberg, M. Sundberg, E. Wang, J. Bergman, and J. Sachs, *Cellular internet of things: Technologies, standards, and performance*. Amsterdam: Academic Press, 2017.

[18] Yate Bts, *LTE Architecture Concepts.* [Online]. Available: https://yatebts.com/documentation/concepts/lte-concepts/#The_MME (accessed: Mar. 28 2021).

[19] *Cellular System Support for Ultra-Low Complexity and Low Throughput Internet of Things: Technical Report 45.820 v13.0.0,* Third Generation Partnership Project, 2016.

[20] *TS 137 104 - V13.3.0 - Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); LTE; E-UTRA, UTRA and GSM/EDGE; Multi-Standard Radio (MSR) Base Station (BS) radio transmission and reception: 3GPP TS 37.104 version 13.3.0 Release 13*.
[21] Z. S. C. Bormann, “RFC 7959 - Block-Wise Transfers in the Constrained Application Protocol (CoAP),”
[22] Eclipse, *Wakaama: LwM2M.* [Online]. Available: https://github.com/eclipse/wakaama
[23] Eclipse, *Leshan: LwM2M.* [Online]. Available: https://github.com/eclipse/leshan
[24] Eclipse, *Paho: MQTT.* [Online]. Available: https://github.com/eclipse/paho.mqtt.c
[25] Eclipse, *Mosquitto: MQTT.* [Online]. Available: https://github.com/eclipse/mosquitto
[26] T. Ekman and S. Jönsson, “An empirical study of Cellular-IoT: Master’s thesis in Communication Engineering,” Master Thesis, Chalmers University of Technology, Department of Electrical Engineering, Gothenburg, Sweden, 2019. Accessed: Mar. 12 2021.
[27] Deutsche Telekom, “Narrowband-IoT Delivers: Insights from the largest NB-IoT indoor measurement campaign,” pp. 1–4, 2019. [Online]. Available: https://iot.telekom.com/resource/blob/data/165170/aedf685e59ad421069657ff7d3408fde/narrowband-iot-delivers.pdf
[28] IoT Creators by Deutsche Telekom IoT GmbH, *Roaming Network Info: European Telekom and Non-Telekom Roaming Partners, Non-European Roaming Partners.* [Online]. Available: https://docs.iotcreators.com/docs/telekom-iot-networks-in-europe (accessed: Apr. 29 2021).
[29] Quectel, *LTE BG96 Cat M1/NB1/EGPRS: Product Specifications.* [Online]. Available: https://www.quectel.com/wp-content/uploads/pdfupload/Quectel_BG96_LTE_Specification_V1.7.pdf (accessed: February 2021).
[30] 1NCE, *All features of the 1NCE IoT Flat Rate at a glance.* [Online]. Available: https://1nce.com/en/features/ (accessed: January 2021).
[31] Wikipedia contributors, *Hayes command set.* [Online]. Available: https://en.wikipedia.org/w/index.php?title=Hayes_command_set&oldid=1018983068 (accessed: Apr. 29 2021UTC).
[32] *BG96 - AT Commands Manual: LTE Module Series,* Quectel, Sep. 2018.
[33] Sixfab, *Technical Details: This section introduces the concepts and terms that are crucial for the understanding, implementation and use of Cellular IoT HAT.* [Online]. Available: https://docs.sixfab.com/docs/raspberry-pi-cellular-iot-hat-technical-details (accessed: Apr. 29 2021).
[34] Qoitech, *Otii Arc.* [Online]. Available: https://www.qoitech.com/otii/ (accessed: 30th April 2021).
[35] Eclipse Foundation, *Repositories.* [Online]. Available: https://github.com/eclipse (accessed: 17th May 2021).
[36] Eclipse Foundation, *Eclipse Mosquitto™: An open source MQTT broker.* [Online]. Available: https://mosquitto.org/ (accessed: 2nd May 2021).
[37] Eclipse Foundation, *Californium (Cf) - CoAP for Java.* [Online]. Available: https://github.com/eclipse/californium (accessed: 17th May 2021).
[38] L. Durkop, B. Czybik, and J. Jasperneite, “Performance evaluation of M2M protocols over cellular networks in a lab environment,” in *2015 18th International Conference on Intelligence in Next Generation Networks*, Paris, France, 2015, pp. 70–75.
[39] E. Rescorla, H. Tschofenig, and N. Modadugu, “The Datagram Transport Layer Security (DTLS) Protocol Version 1.3: draft-ietf-tls-dtls13-43,” [Online]. Available: https://tools.ietf.org/pdf/draft-ietf-tls-dtls13-43.pdf

[40] M. T. Abbas *et al.,* “Guidelines for an Energy Efficient Tuning of the NB-IoT Stack,” in *2020 IEEE 45th LCN Symposium on Emerging Topics in Networking (LCN Symposium)*, Sydney, Australia, 11172020, pp. 60–69.

Abbreviations and Acronyms

ACK	Acknowledgement
ACL	Access Control List
ACO	Access Control Object
AMPQ	Advanced Message Queuing Protocol
APN	Access Point Name
AS	Application Server
AWS	Amazon Web Services
BL	Bandwidth-reduced Low-complexity
BLE	Bluetooth Low Energy
BPSK	Binary Phase Shift Keying
BR	Bandwidth Reduced
BS	Base station
BSS	Base Station Subsystem
BW	Bandwidth
BWT	Block-Wise Transfer
CA	Certificate authority
CBOR	Concise Binary Object Representation
CE	Coverage Enhancement
CFO	Carrier Frequency Error
CID	Connection Identifier
CINR	Carrier to Interference and Noise Ratio
CIoT	Cellular IoT
CoAP	Constrained Application Protocol
CP	Cyclic Prefix
DB	Database
DER	Distinguished Encoding Rules
DL	Download
DNS	Domain Name System
DRNG	Deterministic Random Number Generator
DRX	Discontinuous Reception
DTLS	Datagram Transport Layer Security
DUP	Duplicate Delivery of a Publish Packet
ECDHE	Elliptic Curve Diffie-Hellman Ephemeral
ECDSA	Elliptic Curve Digital Signature Algorithm
EGPRS	Enhanced General Packet Radio Service
EPC	Enhanced Packet Core
EST	Enrolment over Secure Transport
FD	Full Duplex
FDD	Frequency Division Duplex

FEC Forward Error Correction
GNSS Global Navigation Satellite System
GPIO General Purpose Input Output
HD Half Duplex
HSS Home Subscriber Server
IANA Internet Assigned Numbers Authority
ICV Integrity Check Value
IETF Internet Engineering Task Force
IMSI International Mobile Subscriber Identity
IoT Internet of Things
IP Internet Protocol
ISI Inter Symbol Interference
JSON JavaScript Object Notation
LPWAN Low Power Wide Area Network
LTE Long Term Evolution
LVS Linux Virtual Server
MIMO Multiple Input Multiple Output
MME Mobility Management Entity
MO Mobile Originated
MQTT Message Queuing Telemetry Transport
MSR Multi-Standard Radio
MT Mobile Terminated
MTC Machine-Type Communication
MTU Maximum Transmission Unit
MVNO Mobile Virtual Network Operator
NAS Network Attached Storage
NAT Network Address Translation
NIDD Non-IP Data Delivery
NON CoAP Non-Confirmable
OASIS Organization for the Advancement of Structured Information Standards
OFDM Orthogonal Frequency Division Multiplexing
OMA Open Mobile Alliance
OpenAPI Publicly Available Application Programming Interface
OpenSSL Publicly Available Secure Sockets Layer
OSI Open Systems Interconnection model
PA Power Amplifiers
PCRF Policy and Charging Rules Function
PDCHH Physical Downlink Control Channel
PDP Packet Data Protocol
PK Private Key
PKIX Public-Key Infrastructure Working Group
PPP Point to Point Protocol

PRB Physical Resource Block
PSK Pre-shared key
PSM Power Saving Mode
PSS Primary Synchronization Signal
QoS Quality of Service
QPSK Quadrature Phase Shift Keying
QZSS Quasi-Zenith Satellite System
RAI Release Assistant Indication
RAT Radio Access Technology
RAU Routing Area Update
RE Resource elements
RFC Request for Comments
RPK Raw public key
RRC Radio Resource Control
RSRP Reference Signal Received Power
RSRQ Reference Signal Received Quality
RSSI Received Signal Strength Indication
RX Receiver
SCRAM Salted Challenge Response Authentication Mechanism
SIB System Information Block
SIM Subscriber Identity Module
SIP Session Initiation Protocol
SSL Secure Socket Layer
TAU Tracking Area Update
TBCC Tail Biting Convolutional Code
TCP Transmission Control Protocol
TDD Time Division Duplex
TLS Transport Layer Security
TLV Type-Length-Value
TMSI Temporary Mobile Subscriber Identity
UART Universal Asynchronous Receiver-Transmitter
UDP User Datagram Protocol
UE User Equipment
UI User Interface
UL Upload
URI Uniform Resource Identifier
US United States
VM Virtual Machine
VPC Virtual Private Cloud
XMPP Extensible Messaging and Presence Protocol

List of Figures

List of Tables

TH Köln
Gustav-Heinemann-Ufer 54
50968 Köln
www.th-koeln.de

Technology
Arts Sciences
TH Köln