Politecnico di Torino

POLITECNICO DI TORINO

Master's Degree Course in Computer Engineering

Master's Degree Thesis

An Analysis of Possible Architectures and MQTT Communication Protocol Applied to the Connected Vehicles Scenario

Supervisor
prof. Fulvio Risso

Candidate
Cleide Tomaselli

ACADEMIC YEAR 2020-2021

Contents

Abstract

Lately, we have assisted the boom of the so-called "Internet of Things" (IoT): this phenomenon describes a network of physical objects that are embedded with sensors, software and other technologies for the purpose of connecting and exchanging real time data with other devices and systems over the Internet. Connectivity applied to objects of common use is maybe one of the most important technological revolutions of the 21st century, capable of producing numerous benefits both from the business side and for the final consumers, in terms of effectiveness, efficiency and productivity.

Among the many possible IoT devices that can be connected nowadays, this thesis work will focus on the connected vehicles scenario. Following the introduction of a normative that obliges carmakers to install a SIM card and to implement a mechanism to make an emergency call on board, the main OEMs have necessarily had to add specific hardware and software on the vehicle and have therefore decided to exploit this to their own advantage, by installing telematic boxes. These embedded devices were then connected to various control units to obtain useful information about the vehicle, which can then also be sent to the customer's mobile app.

This means that carmakers have seen a big chance in using telemetry data retrieved from vehicles to build a better user experience for vehicles' owners and to offer new dimensions of infotainment. This was a fundamental change of paradigm, because features that were previ-

ously considered as optional are now becoming part of the production chain of the car in all respects. Furthermore, car manufacturers will be able to continuously update the cars with new software and this is a crucial difference from the traditional model of car ownership in which vehicles immediately depreciate in terms of performance and value.

In such a particular scenario, in which the network is not reliable and in which, due to the hardware limitations of the vehicle, the embedded device can not always be connected to the network, the MQTT protocol enters the scene, because it best fits this world. This protocol, which is briefly presented in one of the opening chapters, fits well to a real-time communication model, is capable of transmitting information with a many-to-one paradigm and is well suited for asynchronous communication, which is a classic requirement of the IoT.

This thesis work aims to analyze a real context and was therefore made based on information provided by an important global OEM with offices in Europe and North America. Although the implementation details of the box were not made available together with the complete architectural choices, the data were sufficient to conduct a study of the possible ways in which the MQTT protocol can be applied to the connected vehicle world, with specific regard to the exchange of messages between the vehicle and the cloud and between the mobile app and the cloud.

In particular, after having presented an high level overview of the infrastructure of the connected vehicle platform similar to the one actually used by the company, the operating mechanisms of the embedded telematic box installed on board the car have been analyzed, in order to prove that this kind of IoT devices can take full advantage of some additional features made available by the MQTT protocol.

In the final part of the work, real use cases of adoption of the MQTT protocol on the embedded box and on the mobile app have been presented: these concern the execution of remote operations on the vehicle and the sending of event-triggered data packets towards the application or third parties systems, depending on the considered scenario.

Chapter 1

Introduction

Every year hundreds of millions of new devices are being connected to the Internet. Internet of things (IoT) connects different objects in order to collect and exchange useful data. With this data, IoT aims to provide a system for the monitoring and control of the physical world.[1]

Everything is connected now: not only the devices we carry with us during the day, such as smartphones and tablets or those we wear like smartwatches, but also those that surround our homes, starting with traditional appliances to get to thermostats and alarm systems.

These objects communicate with each other and with other services using various communication protocols for the transportation of sensor or event data. These protocols enable applications to collect, store, process, describe, and analyze data to solve a variety of problems. IoT also aims to provide secure, bi-directional communication between interconnected devices, such as sensors, actuators, microcontrollers or smart appliances, and corresponding cloud services.[2]

Within this thesis work, after an overview of the IoT world, a focus will be made on one of the main communication protocols adopted in this area, namely MQTT. This protocol is becoming the standard for IoT communication, since the characteristics of these devices have

changed the perspective also from the point of view of transmission methods, because they require the adoption of asynchronous but real-time communication, capable of sending information in one-to-many mode.

Due to the many advantages related to making the devices connected, all sectors are moving in this direction and therefore the automotive context is no exception.

Starting from chapter 4 of this work, the concept of connectivity applied to the automotive context will be described and the related notion of "telematics" will be presented also: the combination of these key ideas implies the possibility of sending, receiving and storing information via embedded systems (or boxes), which have the potential to deliver a large amount of data that most OEM companies could leverage.

The new challenges that OEMs are facing will then be presented. Carmakers' interests have indeed switched from just selling cars to providing a set of services to help the driver and to be considered by customers when choosing to buy a car. OEMs' investments are supported by the belief that automotive industry will benefit from connectivity, as regards the driver experience, the sale of the information collected to potential third parties and the maintenance aspect of the vehicle.

In chapter 5, the attention will focus on a proposed connected car architecture inspired by a real scenario and on the reasons why it is possible to use the MQTT protocol for some communication flows, which concern the execution of remote operations and the sending of event-triggered data.

In chapter 6, the mechanisms at the base of the telematic box in-

stalled on board the car will be analyzed: the real application cases presented in the final part of the work, conducted from the company's point of view but with attention paid to providing the best possible user experience, both when he or she is in the car (or is about to get in) and when using a possible mobile app, are aimed at demonstrating whether the obvious constraints of the embedded boxes can be somehow overcome with the use of the MQTT protocol.

Chapter 2

Internet of Things and Industry 4.0

Nowadays it is possible to connect anything, from something as small as a bulb to something as big as an aeroplane. For this reason, experts are expecting around 30 billion connected IoT devices by 2025 and, in this regard, the adoption of IPv6, which should provide enough IP addresses for every device the world is ever likely to need, was also a necessary step for the IoT to scale.

The expression "Internet of Things" was coined by Kevin Ashton in 1999:

> «The IoT integrates the interconnectedness of human culture – our 'things' – with the interconnectedness of our digital information system – 'the Internet.' That's the IoT.»

Unfortunately, it took at least another decade for the technology to catch up with the vision. In fact, the idea of adding sensors and intelligence to basic objects was discussed throughout the 1980s and 1990s, but apart from some early projects progress was slow simply because the technology was not ready. Chips were too big and bulky and there was no way for objects to communicate effectively. Before it finally became convenient to connect devices, processors were needed that

were cheap enough to be almost disposable. The adoption of RFID tags (low-power chips that can communicate wirelessly) solved some of these issues, along with the increasing availability of broadband Internet and cellular and wireless networking.

2.1 Industry 4.0

The IoT was initially most attractive to business and manufacturing, where its application is sometimes known as machine-to-machine (M2M). This explains the use of the expressions "Industrial Internet of Things (IIoT)", "the fourth industrial revolution" and "Industry 4.0", all given to the use of IoT technology in a business setting. The concept is the same as for the consumer IoT devices in the home, but in this case the aim is to use a combination of sensors, wireless networks, big data, AI and analytics to measure and optimise industrial processes (Figure 2.1).

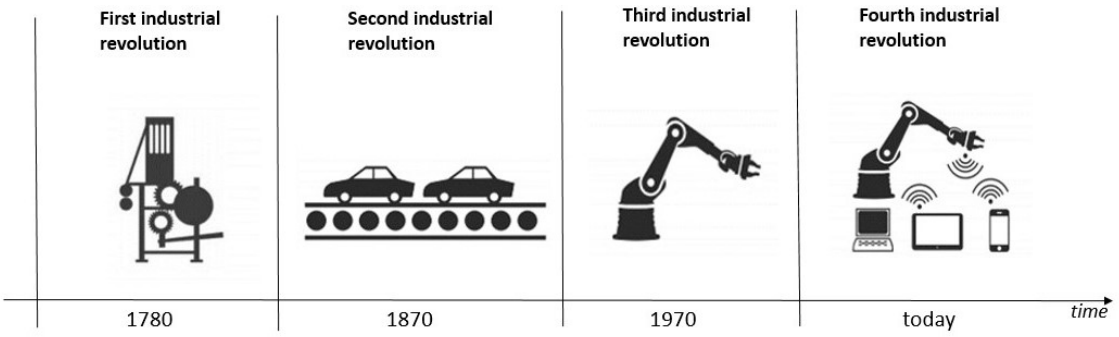

Figure 2.1: Industrial revolutions timeline.

In particular, as microprocessors are now inexpensive and networks are widespread, embedded systems have the potential to deliver a large amount of data that most companies could leverage.

A first example of application in the industry context can be the maintenance of machines. Every machine part has a certain life expectancy: some fragile parts may last a few weeks, some several years.

But sooner or later, physical parts need to be replaced. Most factories rely on their machines running in parallel: once one of them stops, the flow cannot go on. Detecting machine parts that need to be replaced before they actually break can save the company a lot of analysis work and money. With self-monitoring machines, this is about to become more efficient and involve less human maintenance work. By equipping machines with various sensors to run self-tests and verify that all of their parts operate as planned, malfunctions or old parts can be identified early on by the machine. It can then call for a technician to replace part x or manually check part y. By pinpointing possible problems this way, machine downtime can be minimized.[3]

Another example can be found in the airplane industry, where preflight security checks involve a lot of manual work, because each and every piece of essential equipment has to be checked off a list for the flight to be considered safe. On average, this took 6.5 hours per plane. By integrating RFID tags into essential safety equipment, the airplane industry made it possible for security staff to use an RFID scanner instead of a paper checklist to make sure that no important parts are missing. The 6.5 hours it took to manually check for the presence of each part could be reduced to 20 minutes this way, saving a lot of man-work hours.

2.2 What industries can benefit from IoT?

Connected devices combine with automated systems to gather IoT data that can be analyzed to allow certain tasks to be automated, particularly those that are repetitive, time consuming, or dangerous. The benefits of the IoT for business depend on the particular implementation; agility and efficiency are usually top considerations. The idea is that enterprises should have access to more data about their own products and their own internal systems: to pursue this goal,

manufacturers will have to add sensors to the components of their products so that they can transmit data back about how they are performing. This can help companies spot when a component is likely to fail and to swap it out before it causes damages. Companies can also use the data generated by these sensors to make their systems and supply chains more efficient.

Manufacturers can gain a competitive advantage by using production-line monitoring to enable proactive maintenance on equipment when sensors detect an impending failure. Sensors can actually measure when production output is compromised. With the help of sensor alerts, manufacturers can quickly check equipment for accuracy or remove it from production until it is repaired. This allows companies to reduce operating costs, get better uptime, and improve asset performance management.[4]

2.3 IoT usage contexts

IoT technology can be found not only in the industrial field, since the smart home concept is becoming an increasingly widespread reality and the big ambition is assisting the infrastructure of an entire smart city. A set of IoT usage scenarios is listed below.

2.3.1 Smart home

The smart home is one area where the big tech companies (in particular Amazon, Google and Apple) are competing hard. This category includes connected electronic devices that are used in the house and is the most prominent one for end consumers. Some examples of these smart home devices are: smart fridges, smart plugs, smart door locks, smart thermostats, smart scales, smart light bulbs, smart pet food dispensers. These objects often replace traditional electronic devices by enhancing them with an Internet connection and a way to control

them digitally. In most cases, they offer a smartphone app or integration with a smart home app, such as Google Home or Amazon's Echo, which makes it possible, for example, to turn the device on and off, change its settings and check its status. These smart home applications may be able to help keep older people independent in their own homes by making it easier for family and carers to communicate with them and monitor how they are getting on. A better understanding of how our homes operate and the ability to tweak those settings, could also help save energy by cutting heating costs, for example. In fact, using smart meters and a web interface, tenants can check their monthly energy consumption. This way, it is easier to identify electronics or usage patterns that require a lot of energy, just by comparing the monthly costs. There are also home security systems, which make it easier to monitor what is going on inside and outside, or to see and talk to visitors.

2.3.2 Smart cities, transportation and logistics

Looking beyond the home, sensors can help people to understand how noisy or polluted the environment might be. The goal is to save emissions and improve how services are delivered to the public: smart cities could change how we build and manage our public spaces. Transportation and logistics systems can also benefit from a variety of IoT applications. Fleets of cars, trucks, ships and trains that carry inventory can be rerouted based on weather conditions, vehicle or driver availability, thanks to IoT sensor data. The inventory itself could also be equipped with sensors for track-and-trace and temperature-control monitoring. The food and beverage, flower, and pharmaceutical industries often carry temperature-sensitive inventory that would benefit greatly from IoT monitoring applications that send alerts when temperatures rise or fall to a level that threatens the product.[4]

2.3.3 Retail

IoT applications allow retail companies to manage inventory, improve customer experience, optimize supply chain and reduce operational costs. For example, smart shelves fitted with weight sensors can collect RFID-based information and send the data to the IoT platform to automatically monitor inventory and trigger alerts if items are running low.[4] Retailers can strategically place beacons (devices based on Bluetooth technology) around stores in order to push targeted offers to customers who are nearby.

2.3.4 Healthcare

Healthcare devices represent one of the fastest-growing sectors of the IoT market and that is why the expression "The Internet of Medical Things" (IoMT) was coined. The IoMT devices can be used by medical professionals, patients and their caregivers. Remote patient monitoring is the most common application of IoT devices for healthcare. Using connected IoT devices, the healthcare industry can now collect, even from patients who are not physically present in the hospital, metrics like heart rate, blood pressure, temperature and forward them on cloud-based platforms and display them on software applications. IoT devices can also help keep glucose levels under control by providing continuous and automatic monitoring. With a Smart CGM (Continuous Glucose Monitoring) device and an IoT insulin pen, the glucose monitoring and insulin injection can be automated and precise based on the whole day's glucose reading. Additionally, all data are stored online for the patient and their caregivers to check at any time, anywhere. Others interesting examples are the small and very easy to carry devices for monitoring heart rates and the smart oximeter, which send data collected directly to the smartphone's owner.

2.3.5 General Safety Across All Industries

IoT can be used to improve worker safety. Employees in dangerous environments such as mines, oil and gas fields, and chemical and power plants, for example, need to know about the occurrence of a hazardous event that might affect them. When they are connected to IoT sensor–based applications, they can be notified of accidents or rescued as swiftly as possible.[4]

2.3.6 Automotive and smart cars

Another key sector on which the rise of the IoT has a considerable influence is the automotive industry, for which great strides are made every day. IoT applications in this context can be classified as follows:

- impacts on production lines;
- user experience advantages;
- smart car and driving comfort.

The benefits of applying IoT to production lines have already been discussed in the section on industry 4.0, while as regards user experience advantages, just think about how now through a mobile application it is possible to carry out remote operations, such as preheating the car before the driver gets in, or booking a service appointment when the vehicle has signaled that it is time for a check or for doing maintenance. Furthermore, the advancements made in machine learning in the last decade are closely connected to smart cars progress, since this technology is used to develop features like

- detect the street, other cars, and people;
- understand signs and speed limits;
- alert the driver with details and recommendation in case of dangerous situations identified.

The final goal is to create a *mesh* of connected cars. The automotive scenario will be analyzed in chapter 4.

Chapter 3

MQTT: the designated protocol for IoT

The characteristics of the IoT have changed the cards also from the point of view of transmission methods: in fact, these devices require the adoption of a real-time communication model, capable of sending information in one-to-many mode and of receiving an event at the very moment it occurs. Asynchronous communication is a classic requirement of the IoT. For these reasons, one of the most used protocols for the Internet of Things is MQTT, which is becoming as important to IoT developers as HTTP is to web developers.

3.1 The protocol

The Message Queuing Telemetry Transport (MQTT) protocol had its genesis in 1999 thanks to an idea by Andy Stanford-Clark (IBM) and Arlen Nipper (Eurotech). Although it started as a proprietary protocol, it was released Royalty free in 2010 and became an OASIS standard in 2014. MQTT protocol is the communication standard for the Internet of Things.

An excerpt from the ISO standard is reported below:

The expression “Internet of Things” was coined by Kevin Ashton

in 1999:

> «ISO/IEC 20922:2016 is a Client Server publish/subscribe messaging transport protocol. It is lightweight, open, simple, and designed so as to be easy to implement. These characteristics make it ideal for use in many situations, including constrained environments such as for communication in Machine to Machine (M2M) and Internet of Things (IoT) contexts where a small code footprint is required and/or network bandwidth is at a premium. The protocol runs over TCP/IP, or over other network protocols that provide ordered, lossless, bi-directional connections.»[5]

3.1.1 Versions and variants

There are several versions of MQTT:

- MQTT v3.1.0: original not in common use;
- MQTT v3.1.1: in common use;
- MQTT v5: currently limited use;
- MQTT-SN: limited use and still being standardised.

MQTT v3.1.1 is the version most commonly used but there is very little difference between v3.1.0 and 3.1.1. MQTT v5.0 retains all of the features of MQTT v3.1.1 and adds considerably more, like the introduction of reason codes for failure, an expire date field and shared subscriptions to balance the load across clients, but it is not currently in widespread use. MQTT-SN is exclusively designed for sensor networks and therefore has some different features, such as an additional level of QoS, the ability to work on UDP and an auto discovery function.

3.1.2 Features and goals

The main characteristics of MQTT are being a simple and light protocol for exchanging messages, as well as minimizing traffic on networks and requiring few resources from devices for their management. It is also a protocol capable of efficiently performing the distribution of messages from one to many recipients, decoupling the production of the message from the reception of it and scaling systems.

This set of features makes it extremely suitable for all those environments where available resources and network bandwidth are limited, or where there are remote devices with little memory and low computing capacity, making it possible to operate optimally even when dealing with networks that are not perfect in terms of connection stability, subject to multiple interruptions. This protocol is aimed at minimizing device resource requirements (especially in terms of bandwidth) while attempting to ensure reliability and a certain level of delivery assurance. Another important goal is to optimize energy consumption for applications that require battery power.

3.2 The Publish/Subscribe architecture

Instead of the client/server model of HTTP, the MQTT protocol follows a classic "publish and subscribe" one [fig.3.1], which is event-driven, asynchronous, agnostic to the content of the payload of the message and which allows one-to-many messaging. With this architecture, senders will post messages related to a certain topic (for example the temperature) to a server that can handle thousands of clients at the same time. The three actors involved in the model are:

- the **message broker**, which is a server and can be considered as the central hub through which every message must pass: the broker is responsible for receiving and filtering messages, deter-

mining who is subscribed to the matching topic, and sending the message to these clients only. The broker also has the responsibility to authenticate and authorize clients and at the same time can manage the lists of topics in which the recipients are interested and can restrict access to them.

- the **publisher**, which is a client that produces data and send them to the broker in the form of messages related to specific topics.
- the **subscriber**, which is a client that wants to receive messages belonging to the topic (or topics) for which it has made a subscription.

Both publisher and subscriber are MQTT clients that can communicate exclusively through the message broker, so there is no direct connection between them. An MQTT client can be any device, like a sensor, or application that is able to run an MQTT library over a TCP/IP stack and connect to an MQTT broker over a network.

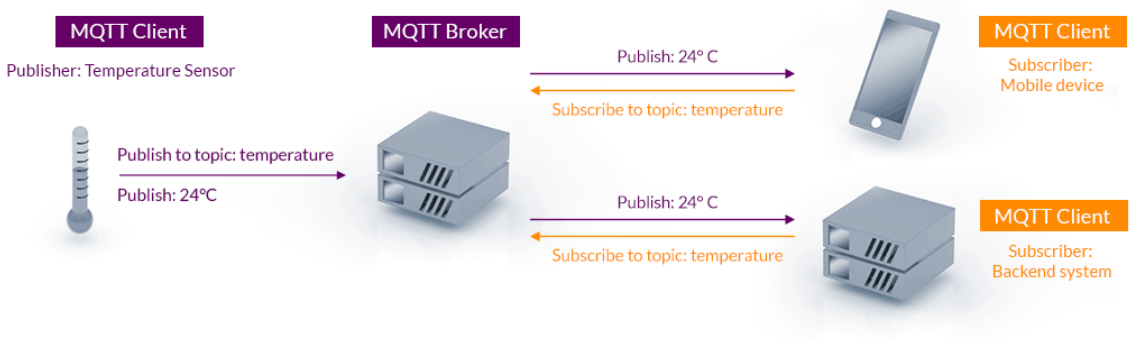

Figure 3.1: An example of publish/subscribe exchanging messages involving a temperature sensor.[6]

.

3.2.1 The topics

The MQTT topic is the routing information for the broker. Topics are UTF-8 strings structured in a hierarchy similar to directories and files in a file system, using the forward slash (/) as a delimiter for levels. A client can subscribe to individual or multiple topics. When subscribing to topics two wildcard characters can be used. They are:

- # (hash character) - multi level wildcard
- + (plus character) - single level wildcard

Topics will be removed from a broker when the last client that is subscribing to that broker disconnects (if the clean session flag is set to true).

Taking a house temperature monitoring system as an example, a possible topic can be "home/bedroom/sensors/temperature".

3.2.2 The most common broker tools

When it comes to MQTT brokers there are three main options: use a locally installed broker/server, use a cloud based server or a virtual server or use a shared server application.

The most popular brokers for local installs are:

- **Eclipse Mosquitto**: it is an open source (EPL/EDL licensed) message broker that implements the MQTT protocol versions 3.1.0 and 3.1.1. Mosquitto is lightweight and is suitable for use on all devices from low power single board computers to full servers. The Mosquitto project also provides a C library for implementing MQTT clients, and the very popular mosquitto pub and mosquitto sub command line MQTT clients. Mosquitto is part of the Eclipse Foundation.

- **HiveMQ**: it is an MQTT broker and a client based messaging platform designed for the fast, efficient and reliable movement of data to and from connected IoT devices. It uses the MQTT protocol for instant, bi-directional push of data between your device and your enterprise systems. HiveMQ meets the requirements of a highly scalable, integratable into backend systems, easy to monitor, and failure-resistant broker by using state-of-the-art event-driven network processing, an open extension system, and standard monitoring providers.
- **Azure IoT Hub**: it is a Platform-as-a-Services (PaaS) managed service, hosted in the cloud, that offers a cloud-hosted back-end solution to connect virtually any device. It provides per-device authentication, built-in device management and scaled provisioning.

3.3 MQTT Message

MQTT messages have the advantage of lightness. An MQTT packet consists of up to three parts, always in the following order: a fixed header (always present), a variable header and a payload.

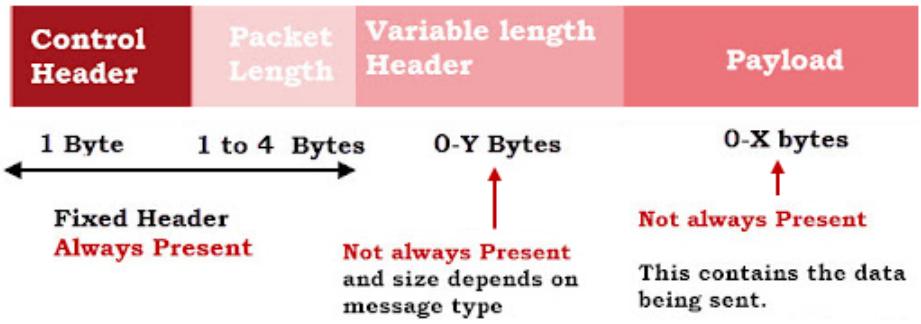

Figure 3.2: MQTT packet format.[7]

The *fixed header* field is divided into control header and packet length fields. The control header is divided into two 4 bit fields and

contains all of the protocol commands and responses. The first 4 most significant bits are for the packet type field and the other 4 bits are used as control flags, specific to each packet type.

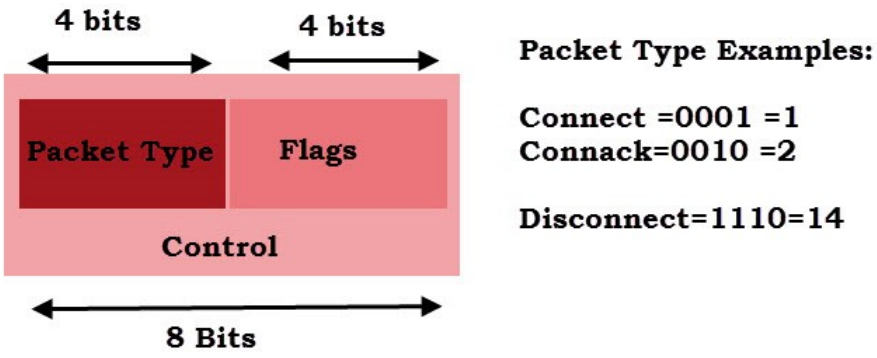

Figure 3.3: MQTT control field structure.[7]

Although there are 16 possible flags, very few are actually used. The most used are:

- The Duplicate flag, used when re-publishing a message with QOS or 1 or 2;
- QoS flags, used when publishing for indicating the QoS level;
- Retain Message flag, also used on publishing.

The presence and the content of the *variable header* depend on the type of packet.

As for the *payload*, it consists of binary data and the format is application specific. Applications can use any data format for the load, such as JSON, XML, encrypted binary or Base64, as long as the recipient clients can parse the payload.

For example, for the three main packets type, this is what happens:

- For a CONNECT packet, sent by the Client to request a connection to a Server
  - the variable header will consists of four fields in the following order: Protocol Name, Protocol Level, Connect Flags and Keep Alive;
  - the payload will contain one or more fields, whose presence is determined by the connect flags in the variable header. These fields, if present, must appear in this order: Client Identifier, Will Topic, Will Message, User Name, Password.
- For a PUBLISH packet, sent from a Client to a Server or from Server to a Client to transport an Application Message
  - the variable header will contain Topic Name and Packet Identifier;
  - the payload will contain the Application Message that is being published. The content and format of the data will be application specific.
- For a SUBSCRIBE packet, sent from the Client to the Server to subscribe to a topic
  - the variable header will contain the Packet Identifier only;
  - the payload must contain at least one Topic Filter/QoS pair. A Topic Filter is a UTF-8 encoded string. A SUBSCRIBE packet with no payload is considered a protocol violation.

3.3.1 Optional functions

When a client publishes a message, the broker usually distributes it to any connected clients that have subscribed to that topic. Once the message has been sent to those clients it is removed from the broker. If no clients have subscribed to the topic or they are not currently

connected, then the message is removed from the broker. In general the broker does not store messages, but retained messages, persistent connections and QoS levels can change this behaviour and the result can be that messages will be stored temporarily on the broker and sent to the affected recipients as soon as they come back online.

To manage the described behaviour, the following fields are used:

- **Clean Session Flag**: MQTT clients by default establish a clean session with a broker. A clean session is one in which the broker is not expected to remember anything about the client when it disconnects. With a non-clean session the broker will remember client subscriptions and may hold undelivered messages for the client. However, this depends on the Quality of Service used when subscribing to topics and the one used when publishing to those topics. Clean sessions are also known as "non-persistent connections" and non-clean sessions as "persistent connections". With the clean session bit it will be possible to control the lifetime of the session state: if is set to 0, the server must resume communications with the client based on state from the current session (as identified by the Client Identifier). If there is no session associated with the Client Identifier, the server must then create a new session. After the disconnection of a session that had the clean session flag set to 0, the server must store further QoS 1 and QoS 2 messages that match any subscriptions that the client had at the time of disconnection as part of the session state.

- **Retain Flag**: it must be set when a client sends a publish kind a message to a broker. It is normally set to 0, which means that the broker does not need to keep the message, but if it is set to 1 then the last message received by the broker on that topic with the retained flag set will be kept. For these cases, the server must store the Application Message and its QoS, so that it can

be delivered to future subscribers whose subscriptions match its topic name. Retained messages do *not* form part of the session state in the server, so they must not be deleted when the session ends. The main use of this flag is for sensors that do not change state very often and publish their status infrequently. This will allow new subscribers to receive the most recent state.

Another interesting optional function allows subscribers to be be notified if the publisher is unavailable, for example due to an I/O error or network outage, and this is realized thanks to the **Will Flag**, which is associated to the topic. The flag enables the sending of a Last Will Message, stored on the broker and sent to any client that has subscribed to that topic if the connection to the publisher fails. If the publisher disconnects normally, the Last Will Message will not be sent.

3.3.2 QoS (Quality of Service) levels

Reliability of message delivery is important for many IoT use cases. This is why MQTT has 3 defined Quality of Service levels, which must be specified from the client:

- "At most once", where messages are delivered according to the best efforts of the operating environment. Message loss can occur. This level could be used, for example, with ambient sensor data where it does not matter if an individual reading is lost as the next one will be published soon after. There is no type of confirmation of receipt, so it does not guarantee message delivery. This is the default one and is called "**QOS level 0**".

- "At least once", where messages are assured to arrive but duplicates can occur. This is called "**QOS level 1**".

- "Exactly once", where messages are assured to arrive exactly once. This level could be used, for example, with billing systems where duplicate or lost messages could lead to incorrect charges being applied.[5] There is a confirmation of receipt, so this guarantees message delivery with no duplicates. This is called "**QOS level 2**".

A different level can be specified for each client/broker connection. Please note that if a client wants to ensure that subscribers get a message even though they might not be online, then it needs to publish with a quality of service of 1 or 2.

3.4 Security

As a transport protocol, MQTT is concerned only with message transmission and it is the implementer's responsibility to provide appropriate security features. This is commonly achieved by the broker using TLS and the recommended TCP port 8883. It is important to note that some security mechanisms are implemented and configured on the broker side, but it is the client that then needs to comply with them.

For example, there are three ways for a broker to verify the identity of an MQTT client, listed from least secure to safest:

1. All MQTT clients must provide a **client id**, which must be unique. When a client subscribes to a topic the client id links the topic to the client and to the TCP connection. With persistent connections the broker remembers the client id and the subscribed topics.

2. An MQTT broker can require a valid **username and password** from a client before a connection is allowed. The username/password combination is transmitted in clear text and is

not secure without some form of transport encryption. However, it does provide an easy way of restricting access to a broker and is probably the most common form of security used.

3. **x509 Client Certificates** is the most secure method of client authentication but also the most difficult to implement because it requires the deployment and management of certificates on client's side. This form of authentication is used for a small number of clients that need a high level of security. However, since MQTT protocol is not trust symmetrical and it provides no mechanism for the client to authenticate the server, the certificates can also be used in reverse.

Server implementations might also monitor clients' strange behavior (like repeated connection attempts, abnormal termination of connections, attempts to send or subscribe to many topics) to detect potential security incidents and might disconnect or blacklist clients that breach their security rules.

As regarding the protection of the MQTT messages, there are three possible options to follow:

1. Adopt **TLS/SSL security**, which is the technology that provides an encrypted pipe down which protects all parts of the MQTT message and not just the message payload. The problem with this solution is that it requires client support, and it is unlikely to be available on simple clients.

2. Adopt **payload encryption**, done at the application level and not by the broker. This means the possibility of having encrypted data without having to configure the broker. However, this type of encryption does not protect passwords (if used) on the connection itself. Because this option does not involve any broker configuration or support, it is likely to be a very popular method of protecting data.

3. The use of **VPNs** to connect clients and servers, in addition to providing confidence that clients are connecting to the intended server, can provide integrity of data across the section of the network covered by a VPN.

3.5 Pros and cons of the protocol

In summary, the MQTT protocol has numerous advantages that motivate its widespread application in the IoT context and the real possibility to scale in the next future to connect billions devices. First of all, it is a lightweight, flexible (it can support different application scenarios), fast, reliable (thanks to QoS levels) network protocol, designed for situations where low consumption is required and where bandwidth is limited.

The real strength of the protocol is the publishing/subscribe architecture because it provides the possibility to access and manage in real time any data that passes through the broker and at the same time it guarantees one-to-many communication, decoupling the production of the message from the reception of it.

The main cons, or rather the main concerns, involve two aspects: first of all, it must be considered that MQTT was born for machine to machine communication, while today there is an increasing need for machine to human communication, because the data generated by sensors often have a human being as their recipient. The second one has to do with still too many safety aspects to consider when implementing or using MQTT: a recent research conducted by the security vendor Trend Micro, conducted over a period of 4 months, showed that over 200 million MQTT messages had been stolen due to exposed servers. Attackers can thus remotely control IoT endpoints or carry out DDOS attacks.[8]

3.5.1 Comparison with HTTP protocol

HTTP (HyperText Transfer Protocol) is the information transmission protocol used on the Web. While it makes sense to question why it is not adopted in the IoT world, the honest answer is that HTTP is not really suitable for this context. In fact, if the IoT devices will be managed through this protocol, the process would involve sending data on the client's HTTP request and receiving updates from the system as soon as an HTTP response is obtained and this request/response model would encounter many limits rather strict due to the nature of the protocol itself:

- HTTP is a synchronous protocol, where the client has to pull the information it needs from the server. This would result in poor scalability that would not be acceptable in the IoT world, where the large number of devices and an unreliable or high-latency network make synchronous communication problematic. Therefore, an asynchronous protocol like MQTT is much more suitable for IoT applications and services. The sensors can send their readings and let the network identify the optimal route and timing for delivery to the recipient devices and services.

- HTTP is unidirectional: it must be the client who initiates the connection. In IoT applications, devices or sensors are normally clients, which means they can not passively receive commands from the network.

- HTTP is a 1-to-1 communication protocol: the client makes a request and the server responds. This makes it difficult and expensive to transmit a message to all the devices connected to the network, which is a very common case for IoT applications.

- HTTP is a heavy protocol with many rules (it is considered by experts to be "verbose", since it always involves sending many

bytes), certainly not ideal for machine-to-machine communication, generally characterized by the emission of a limited number of bits.

Chapter 4

Automotive and connected vehicles

The automotive industry is dealing with massive challenges due to new regulamentations in terms of emissions and this has resulted in a particular attention to electrification, but in addition to this at the same time there has been another big change: the creation of a connected car model. In fact, carmakers have seen a huge chance in using telemetry data from vehicles to create new revenue opportunities and to build a better user experience for vehicles' owners.

Over the last decade, this industry has benefited from tremendous digital innovation, with sophisticated autonomous and assisted driving, proactive vehicle monitoring and maintenance, fleet and traffic management, and safety and infotainment capabilities. With these innovations, vehicle designs are becoming increasingly software-centric and dependent on connectivity and cloud and edge computing capabilities. [9]

4.1 New challenges

The connected car allows manufacturers and dealers to reverse the car ownership model: previously, manufacturers had an arms-length relationship with individual buyers, for which the manufacturer's re-

lationship with the car ended once it was sent to the dealer. Now, progress made in the IoT's world allow manufacturers to upgrade their cars continuously with new software and this is a crucial difference from the traditional model of car ownership in which vehicles immediately depreciate in terms of performance and value. So carmakers' interests have switched from just selling cars to providing a set of services/features. This translates into the choice to automate certain tasks that are repetitive, time consuming and dangerous and to implement new features like usage based insurance, predictive maintenance, remote controls and remote tracking of cars. These services are considered a way to provide added value to customers and are a factor to consider when choosing to buy a car.

Today's cars are asked to relieve drivers from the most stressful operations needed for driving, providing them with interesting and updated entertainment functions. In the meantime, they have to comply with increasingly stringent standards about safety and reliability.

Another important challenge is mobility: trends show that customers are not interested anymore in owning a car, due to the general increase in the cost of purchasing and maintaining a vehicle. For these reasons, carmakers have been also focusing on providing mobility solutions, like car sharing and leasing purchase options, for this set of customers. It should not be forgotten that connected vehicles are part of a bigger ecosystem [fig.4.1], in which the use of new digital technologies is changing the way people move. A key concept is in fact that of **Mobility as a Service** (MaaS), for which the mobility to be guaranteed to the citizen is independent of the means of transport used, as long as efficiency and sustainability are ensured. The innovative concept of MaaS goes in the direction in which mobility will no longer be private (the citizen's personal car) but integrated into a model that puts travelers in the center of a whole connected sup-

porting infrastructure, which gives them the opportunity to choose from several available options. This concept is born for the application mainly in the big cities where traffic and pollution have reached their peaks.

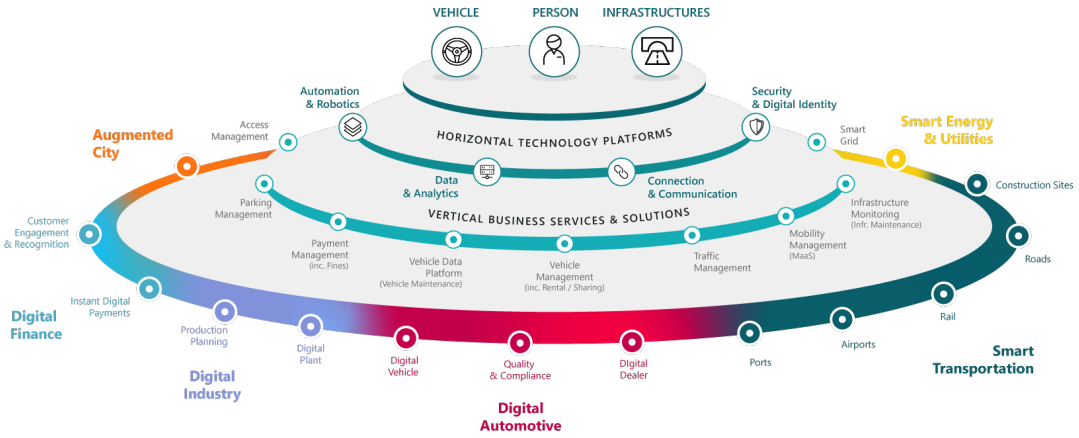

Figure 4.1: Digital ecosystems.[14]

Summing up, it is clear that connectivity will become a key enabler of mobility services, on which are foreseen huge investments in the next decade from OEMs (Original Equipment Manufacturer). Investments are supported by the belief of automotive industry in the growth of business related to connected services, since a connected vehicle gathering data that can be interpreted and leveraged will not only enhance purchasing, driving experience and maintenance of the vehicle itself, but combined with smart telematics and cloud technology will also transform the entire automotive industry into one of automation and self-sufficiency.

In other words, even if carmakers are still keeping the vehicle at the centre of their business model, they are moving in the direction of owning the customer experience, and connection is the key to develop a strategy helping to solve customer requirements while also generating new revenue resources.

4.2 Connectivity

The concept of connectivity is strictly connected to what until recently was referred to as "**telematics**", which indicates the possibility of sending, receiving and storing information via telecommunication devices in conjunction with effecting control on remote objects. Nowadays, two different types of connectivity can be defined:

- **Tethered**: it consists in using a smartphone or a dongle device to provide connectivity to the vehicle. Smartphones are exploited by mirroring interfaces in order to bring an experience user is familiar with. Dongle devices are something users do not usually interact with, but are used for tracking purposes such as usage based insurance. In this solution, to make the connection between the car and the brought-in device, Bluetooth or Wi-Fi can be used.

- **Embedded**: it is the way through which the vehicle manufacturer integrates the information and entertainment platforms into the fundamental design of the car. The majority of OEMs have adopted this approach as it is likely to become the preferred mode of connectivity for the industry in the future.[10] This solution involves a SIM card installed during the manufacturing process and which allows the identification of individual vehicles. This technique has the advantage of relieving the driver of the need to possess a SIM specific for the car and provides the best communication performance because it is reliable since it uses the vehicle antenna and is consequently preferable when safety-critical services are taken into account.[11]

Given this two different modes of integration on board, a connected car is defined as a vehicle which is:

- capable of accessing the Internet at anytime, using either a built in device or brought in user devices;
- equipped with a set of modern applications and dynamic contextual functionalities, offering advanced infotainment features to the driver and passengers;
- capable of interacting with other smart devices on the road or in mechanical shops, leveraging vehicle-to-road infrastructure communication technologies;
- capable of interacting with other vehicles, leveraging vehicle-to-vehicle communication technologies.[11].

4.3 The connected car ecosystem

A connected car ecosystem needs to contain the following elements:

- **Infotainment system**: it is placed in a central position on the control panel. It represents the dashboard through which the driver and the occupants are served with information related to their journey (such as maps, weather, attractions, places of interest, congested roads and nearby accidents) or with entertainment options (such as satellite radio, music, internet connectivity and social networking applications). In particular, the most modern systems are integrated with Apple and Android software to ensure that smartphones can interact better with on-board electronics and allow for optimized management of some apps and contents from the car display. For example, it will be possible to check the calendar and read emails while driving.
- **Cloud**: it is the means that allows management of data and remote commands from and to the vehicle. Once data are gathered from the vehicles, cloud enables data analysis from the vehicle.

Data analysis can be used in several ways, to benefit both carmakers and customers. Carmakers take advantage as they are able to access huge amounts of data for improving the product or making profit by selling data itself. Customers can benefit as those data are used for monitoring the health of vehicle, to notify for maintenance or to access services for a stolen case.

- **Mobile app**: always more often, cars are launched on the market with a related smartphone app, which represents an interface through which user is enabled for a remote monitoring of his/her vehicle: controlling statistics, fuel left and distance to next service are made easier through information brought by cloud from vehicle and displayed on mobile app. Mobile application is gaining importance mainly because of the advent of electrical vehicles that have different needs in terms of life cycle management, as it allows control and monitoring of energy flows and charging sessions. Finally, mobile app represents a touch point through which customers are able to purchase both connected and traditional services.

- **Connected ECUs**: Electronic Control Unit represents the physical object that allows the car to communicate with cloud. It can be embedded in the vehicle or unbundled, sold separately from the auto. Telematic boxes are usually integrated with car network and other ECUs for gathering data from vehicle itself. In other words, through intra-vehicle connectivity (meaning the information transmission between the ECUs and sensors disseminated inside the auto), connected ECUs allow the gathering and analysis of data from vehicles. Data acquisition can take place on either wired (e.g., on the CAN bus) or wireless networks.

- **Proprietary Backend systems**: this category includes all internal systems that collect data and make them available to in-

ternal or external services.

4.3.1 Connected services

The services linked to connected vehicles can be split into *customer-based* ones, focused on support the driver, and *vehicle-based* services, focused on car performance. The first category mainly refers to features that allow OEMs to analyze the driver behavior on the basis of data collected from the vehicle itself. On the other hand, as regards vehicle-based services, the examples are numerous and have impacts on the following contexts:

- **Maintenance**: OEMs can monitor vehicle health and predict the need for replacement in advance, notifying users of possible breakdowns and inviting them to the closest service centre. For example, if a person is driving and sees the engine fault light turn on, his/hers connected car can check the sensor and communicate with others sensors in the vehicle before sending data to the manufacturer, who can then offer an appointment to fix the fault at the nearest dealer and ensure that the required replacement parts are in stock, ready for when the client arrives.

- **Software updates**: OTA (Over The Air) features allow OEMs to provide upgrades for enhancements or software patching purposes based on new development and customer requirements. OTA can be divided into FOTA (Firmware Over The Air) or AOTA (App Over The Air).

- **Safety and assistance**: these services are linked to the safety of driver and passengers and are dued to the European **"Ecall regulation"** of 2018, for which OEMs are required to equip new vehicles with a button that will trigger an emergency call to 112 (or to a call center first) in the event of an accident. There are

also other kinds of services, which provide remote assistance to customers in case, respectively, of unexpected car breakdown occurrences or in case of theft of the vehicle.

- **Fleet management**: connectivity is a key factor for fleet management solutions for customers who own large numbers of vehicles, like in the car sharing case.
- **Journey enhancement**: these services provide support for activities like parking and navigation, allowing customers to know the fastest way to the destination with respect to traffic conditions and identifying parking spots in real time.
- **Research and development**: every connected vehicle produces data that, shared with R&D teams, can help OEMs improve back end development and design for the future generation of vehicles.

However, implementing a connected car system that scale to support millions of cars can present some challenges, because for most services there is a requirement for bi-directional communication between the car and the cloud. Cars will send telemetry data to the cloud and enable apps like predictive maintenance, assisted driving, etcetera and, similarly, the car needs to be able to receive messages from the cloud to respond to commands like lock/unlock doors and remote activation of horn or lights.

In the next chapters of this thesis work the attention will focus on real cases of application of the principles of IoT connectivity on embedded systems (the vehicle ECUs) and on architectures and protocols used for communication.

Chapter 5

The architecture of a real scenario

By 2025, every new vehicle will be connected. Estimates predict that vehicles will send 1 to 10 exabytes of data traffic per month to the cloud, at least 1000 times larger than the present volume. This data is valuable for many automotive digital services, such as those that will drive new insights into global fleet operations, uncover new vehicle revenue streams, and improve vehicle, driver and operational efficiencies. Automotive digital services will fuel the connected vehicle market, which is forecast to exceed USD 150 billion by 2025.[9]

To offer solutions up to market expectations, most OEMs and Tier 1 suppliers have launched or are developing their next generation connected car platforms. In this context, the connectivity project of an important global OEM with offices in Europe and North America takes place. The program aims to connect all vehicles by 2022 and the analysis described hereafter has been carried out thanks to the data made available by the company, which from now on will be referred to as "*OEM_X*".

In this chapter, the key concepts of a possible connected car infrastructure, similar to the one of OEM_X, will be highlighted and analyzed.

5.1 A service oriented architecture

The OEM_X solution is based on a Service Oriented architecture (SOA), which is an architectural style with an approach to software development that takes advantage of reusable software components, called "services". Each service is composed of the code and data integrations required to execute a specific business function.[12]
In a SOA, the service is a discrete unit of functionality that can be accessed remotely and acted upon and updated independently. A service presents a simple interface to the requester that abstracts the underlying complexity acting as a black box. In addition, there is the division into microservices to build individual applications in a way that makes them more agile, scalable and resilient. Microservices are a cloud-native architectural approach, the benefits of which are to provide the ability to update code and scale components more easily, reducing the waste and costs associated with scaling the entire application, due to a change linked to a single feature.

5.2 OEM_X architecture

The aim of this section is to provide a high level overview of the infrastructure of the connected vehicle platform of OEM_X company. First of all, the actors involved are listed below, divided into on-board and off-board sides and customer channels.

5.2.1 On-board

The *embedded* on-board components are two:

- The telematic box
- The Head Unit

The telematic box represents the **connected ECU** (Electronic Control Unit) of the vehicle, being equipped with a *modem*. It is integrated via CAN (Controller Area Network) with other ECUs. CAN network enables in particular two ways of communication:

- reading: for features requiring data stream, the box is able to read data signal from CAN bus directly;
- writing: for features as the remote commands, by publishing on CAN bus, the box is able to interact and wake up other ECUs, like the BCM (Body Control Module).

The operating mechanisms of the box will be detailed in paragraph 6.1.

The Head Unit module is placed in a central position on the dashboard and gives the user control over the vehicle's infotainment. There are different models of head units, but the majority of them are based on the Android Automotive system. The head unit is integrated with the telematic box through a CAN network for running the *apps* that are distributed and it communicates directly with the box modem (in order to exchange data with the backend) via USB for flows of data, such as MOTA (Map Over The Air) or AOTA (App Over The Air) that interest head unit only. The Head Unit and the telematic box are connected:

- via CAN, for running the distributed apps;
- via USB, to manage flows of data, such as MOTA (Map Over The Air) or AOTA (App Over The Air).

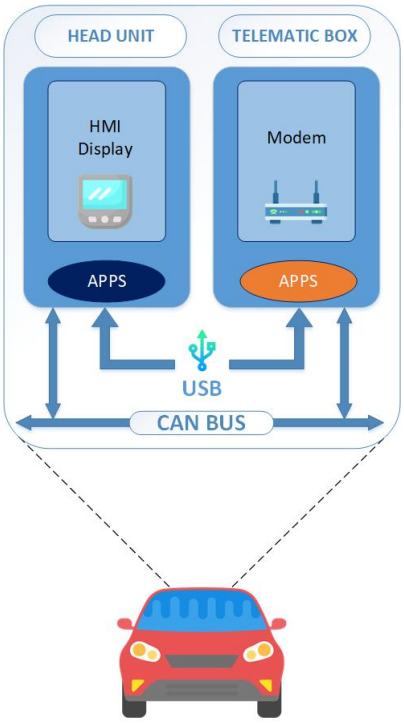

Figure 5.1: A schematic view of the vehicle on-board components.

5.2.2 Off-board

"Off-board" refers to the entire cloud platform on which the connected vehicle leans. It can include both systems hosted on an OEM_X proprietary infrastructure and systems hosted on infrastructures such as those provided by Amazon or Microsoft. The main components are three:

- Backend layer
- Custom layer
- Platform Router

The *Backend* is the first layer that the box may contact and exchange information with, concerning any business case it realize and forwarding information to the other backend stakeholders involved. It allows the customer to use the connected services, even when he or she is out of the car. It represents the closest layer to the vehicle and

is the component in charge of device management through the MQTT standard, data ingestion from vehicle and operation execution on the auto through V2C APIs, the standard interfaces between vehicle and platform.

It can include several systems:

- MQTT broker: it is the component in charge of device management and handles the connection to single topics of the devices.
- Messaging cluster: it is the layer that receives data by MQTT broker according to the topics interested by data flows. It is also in charge of applying some business logic depending on vehicle attributes.
- Stream processor: it represents the component that performs the processing of data published by vehicle and delivered by the messaging cluster according to the different applications requesting the data.
- Database: it stores the information regarding records of each vehicle, also called "vehicle profile".
- Notification Center: it is the component applying the business logic for delivery of notification through user channels to customers.
- FOTA Server: it is the server in charge of the management of the Firmware over the air campaigns.

The *Custom layer*'s duties are to store user information (such as customers' consents), to perform enrichment of data and to define vehicle capabilities for all the channels. It can also represent an Identity Provider.

Finally, the *Platform Router* acts as mediation layer and as API Gateway of the platform, delivering the security, caching and orches-

tration functionality needed to deploy a core API architecture. It represents the logic layer used to communicate with user channels.

5.2.3 Customer channels

The customer channels for OEM_X are the points of contact that the vehicle's owner can use to interact with the vehicle when he or she is not in the car or to get some useful information.

For example, through a mobile app, the owner of an electric vehicle can monitor statistics such as battery status and remaining kilometers before the next recharge. Another interesting use can be pre-heating or defrosting the windshield before getting on board or requesting and scheduling a repair at the trusted dealer. Furthermore, a user who wants to rent a vehicle, also therefore needs to use a fleet app to interact with it.

The web portal, on the other hand, is a channel that allows the user to consult periodic reports on the state of health of the vehicle and to access documentation regarding purchase, guarantee, assistance and supplies. Moreover, through the website or by e-mail, the user can have access to personalized offers and can be invited to specific events organized by the OEM.

The high-level diagram presented in fig.5.2 shows the described OEM_X like architecture.

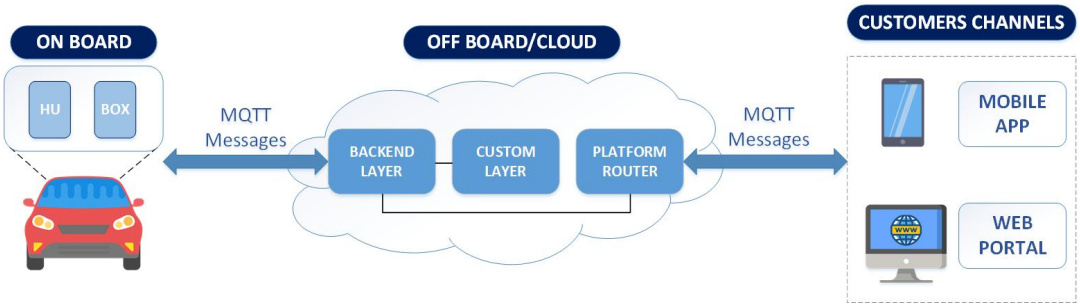

Figure 5.2: Connected vehicle architecture overview.

In addition to these components, a complete ecosystem could also include other external systems to be integrated with the ones described before, like a customer identity management system and a component for handling subscription and billing of services to customers, but this level of detail is out of the scope of this thesis's work.

5.3 The adoption of MQTT protocol

Since HTTP is able to handle sending vehicle telemetry data to the cloud but a request/response model is not good for managing cloud to vehicle messaging, for the exchange of messages for the architecture of the connected vehicle of OEM_X, the best choice is to use a publisher/subscriber model. The latter, as previously explained, better suits such scenarios that include data streaming, sending notifications and event-triggered data.
The communication protocol used for the messages exchanged between the telematic box and the cloud (and between the cloud and the mobile app also) is **MQTT version 3.1.1**. Thanks to MQTT protocol, each vehicle will be decoupled from other vehicles and back-end services and a persistent, reliable and always-on push connection to the cloud will be enabled.

When a network connection is available, a vehicle will publish data to the MQTT broker (which is placed in a component on cloud) and will receive subscribed data from the same broker in near real-time. The broker will perform a store and forward function to route messages from publishers to subscribers. In addition, the broker may prioritize messages in a queue before routing. While the vehicle is offline, the MQTT broker will buffer data and as soon as the vehicle will be back online it immediately will deliver the data. For High Priority messages an SMS may be sent to wake the vehicle. Furthermore, since there is the potential to receive a duplicate message, clientId and messageId

fields will be used to identify and reject any duplicate messages.

5.3.1 V2C (Vehicle to Cloud)

Since OEM_X is a distributed organization, the exchange of MQTT messages is done by means of REST API of V2C kind. V2C (Vehicle to Cloud) is a subset of V2X, which stands for "vehicle to everything", meaning a communication of information between vehicle and every entity that can influence the vehicle and vice versa, including also Vehicle-to-Vehicle (V2V), Vehicle-to road and Infrastructure (V2I) and Vehicle to-Pedestrian (V2P).[13]

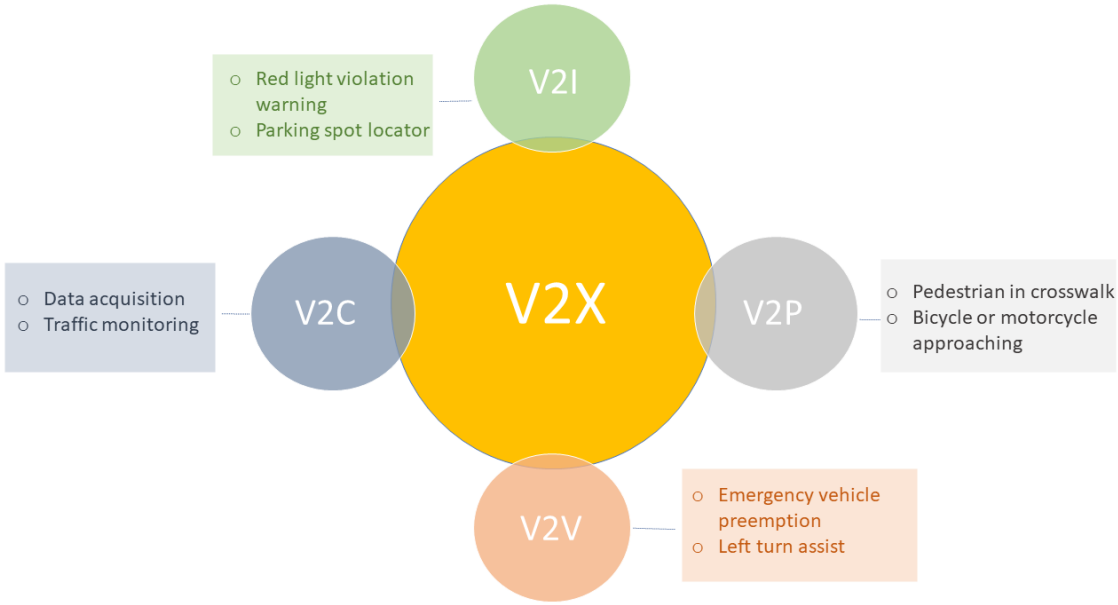

Figure 5.3: V2X map.

OEM_X has defined a specification for data exchanged between the ECUs and the cloud, which is based on industry best-practices for the IoT and in addition to MQTT and HTTP protocols also adopts protocol buffers for efficiently formatting the data into messages. The standard was born out of the necessity to provide a common interface to be used on a global basis, rather than continually adapting the ECUs to interface with the different providers' interfaces.

Chapter 6

Study cases

After having described the complete architecture in a simplified way in the previous chapter, starting from the diagram of fig.5.2, the aim of this chapter will be to explain how MQTT protocol can be applied to the scenario of the OEM company taken in analysis.

The complete MQTT flow can be schematized as follows: Telematic Box and Head Unit on board of the vehicle use the protocol to talk to the first entry point of the OEM_X backend, transmitting data that will be enriched (in case of need) and verified through other internal systems of the OEM_X (after having been crossed with other info) that are on the cloud also. Then, after aggregating the data with other data and after the APIs' calling, MQTT will come back into play to trigger updates to the mobile application.
The flow can also be seen in the opposite direction, if the customer triggers an action on the vehicle from the mobile app (for example, the user can press a button to unlock the doors of a fleet vehicle) and therefore data start from the mobile app and they must reach the vehicle (BCM component included).
Furthermore, in the scene there are also some third parties involved, for example call center platforms in charge of managing certain cases, such as those related to the SOS and stolen vehicle features.

For convenience, the complete flow has been partitioned and analyzed starting from the two ends of the chain. Two different sub-flows of protocol application will be presented: the box-cloud one (head unit flows also pass through the box) and the app-cloud one together with their respective real use cases.

6.1 The telematic box

The telematic box is an IoT device containing a Subscriber Identity Module (SIM) to allow the vehicle to get connected to the 3G/4G network. It communicates with cloud through the MQTT protocol. Data are published on topics and managed by the MQTT broker of the platform, which is in charge of handling messages only to subscribers. Messages are published and retrieved according to APIs. Since the OEM_X architecture provides microservices (or features), every time a new feature comes into play, new topics will be established.

The box speaks MQTT mainly for four reasons:

- MQTT is the protocol to use when the connection is unstable.
- The nature of MQTT is suitable for situations in which, due to hardware limitations, the IoT device will not always be connected: the protocol provides strategies to recover the connection as soon as it is available. With the **clean session** flag set to false it will be possible to keep the undelivered messages queued in the broker, which will deliver them once the client has reconnected.
- The **retain flag** set to true can be useful to keep track of a last known state of the vehicle, for example the position [fig.6.1]: if a client needs to know the latest vehicle position and subscribes to the interested topic, it will automatically receive the information instead of waiting for the device to publish an update, because the position will be already stored and retained inside the topic.

- MQTT is agnostic, so it is possible to exchange messages with multiple payload formats.

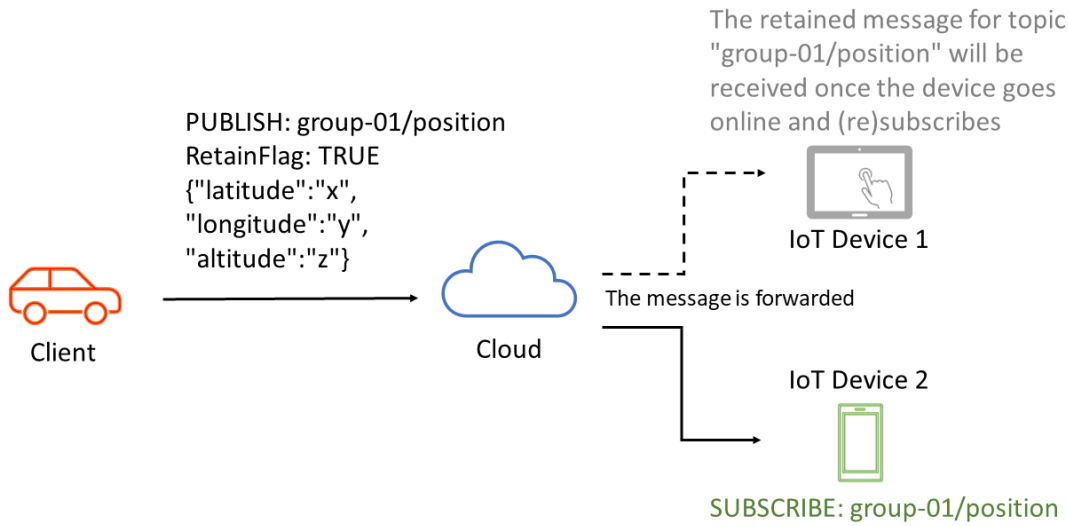

Figure 6.1: Example of position retreiving by using MQTT retained messages mechanism.

Due to the constraints of the vehicle, which does not have infinite resources and cannot always be connected (otherwise it would consume too much), many strategies related to energy consumption can be adopted, such as establishing states for the box. Telematic box states can be defined according to subscription to topics, network availability and energy consumption. Specific triggers can define the status changing.
An example [fig.6.2] of possible states for the box can be the one proposed below:

- **ON**: CAN on triggers this status;

- **LISTEN**: This status is entered when the CAN is off. During this state, the system performs the minimum activities to maintain the connection alive with the server and react immediately to

server notifications; this means that data connectivity is present and MQTT connection is maintained on (through a keep alive data message). This status has a timeout: when it expires or if data connectivity is not present, the box will enter SLEEP mode.

- **SLEEP**: This status is similar to the previous one, but in this one MQTT is disabled, even if the box can be awakened by SMS and after that MQTT connection can be established again.

- **OFF**: This status follows the SLEEP one. In this state neither SMS over network nor data over MQTT topics can wake up the box, which can be only turned up by CAN network in ON state. The box enters in this status after a set number of days of inactivity.

- **LOGISTIC**: Logistic Mode is a working condition common to all the ECUs in the car, useful to guarantee the battery status until the delivery to the consumer and during which the SIM cannot be activated because the ECU is in sleep.

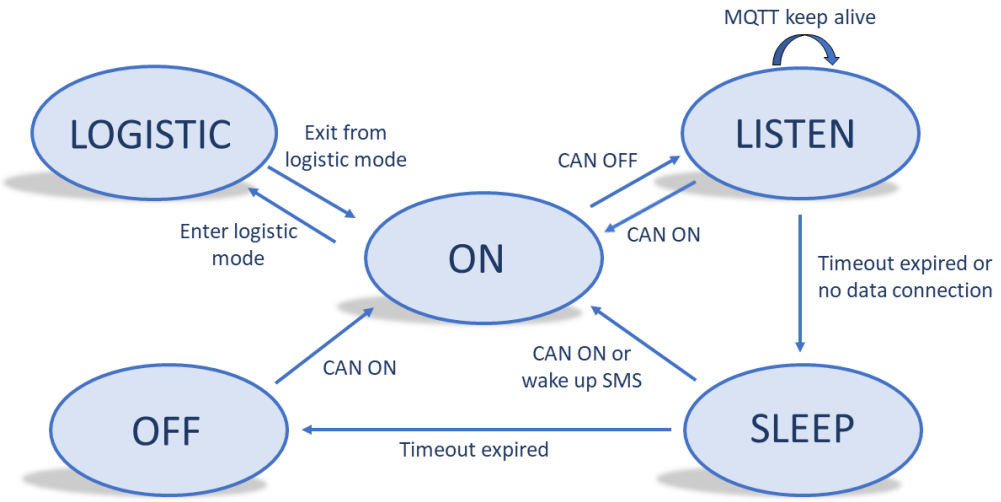

Figure 6.2: The status diagram of the box for the proposed example.

6.1.1 Telematic box-cloud: MQTT flows

Inside the telematic box there is a Telematic Client (TC) component in charge of managing the connectivity and the transmission of MQTT messages and data between the vehicle and the cloud platform.
TC is a background process running within the application framework and shall be started at the boot up of the box, before any other application that needs to communicate with the cloud. The Client will be able to adapt the QoS level on the basis of the operations to be processed.

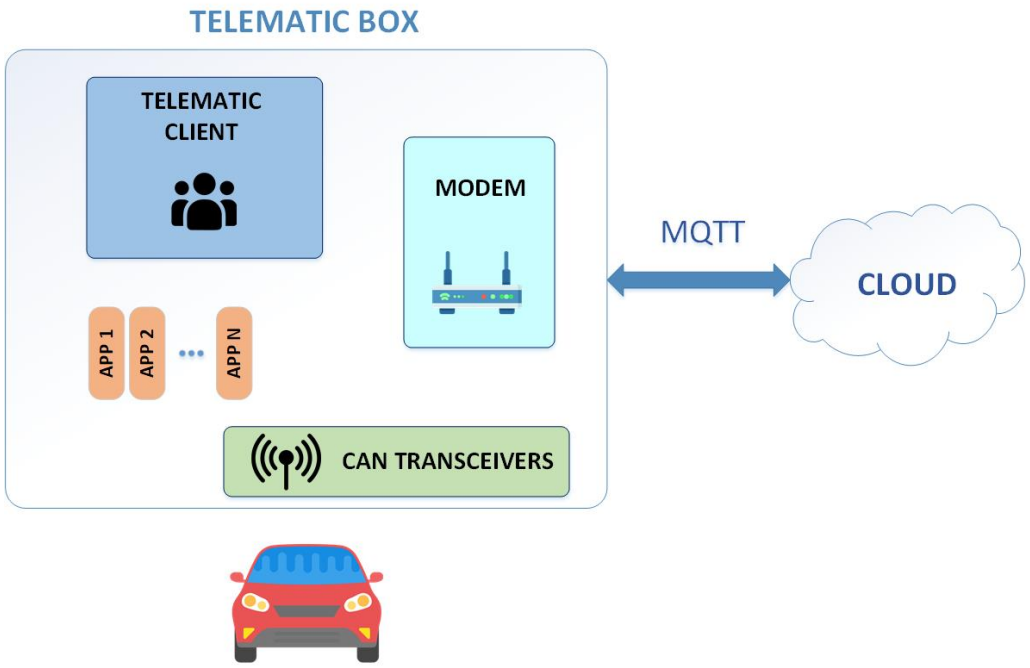

Figure 6.3: Telematic Box - Cloud flows.

Data exchange can be initialized by both sides: if the server wants to send data, it will post messages in a topic for devices that have subscribed to that topic. Technically, TC will subscribe to topics on behalf of the applications that want to make subscriptions.

In the case of subscription to a topic, this can be an example of pseudo code called by an application [fig.6.4]:

- *tClient.subscribe(appID, topicName, topicName_callback)*

When the MQTT broker publishes an event to the interested and subscribed topic, it will be the Client that receives the event first and then calls the callback function to pass the data to the application. This can be an example of pseudo code called by TC:

- *app.topicName_callback(v2c_data)*

In the case of publication of a message, the application will be responsible only for generating the specific V2C data and after that it will be the TC that take care of transmit it to the cloud.

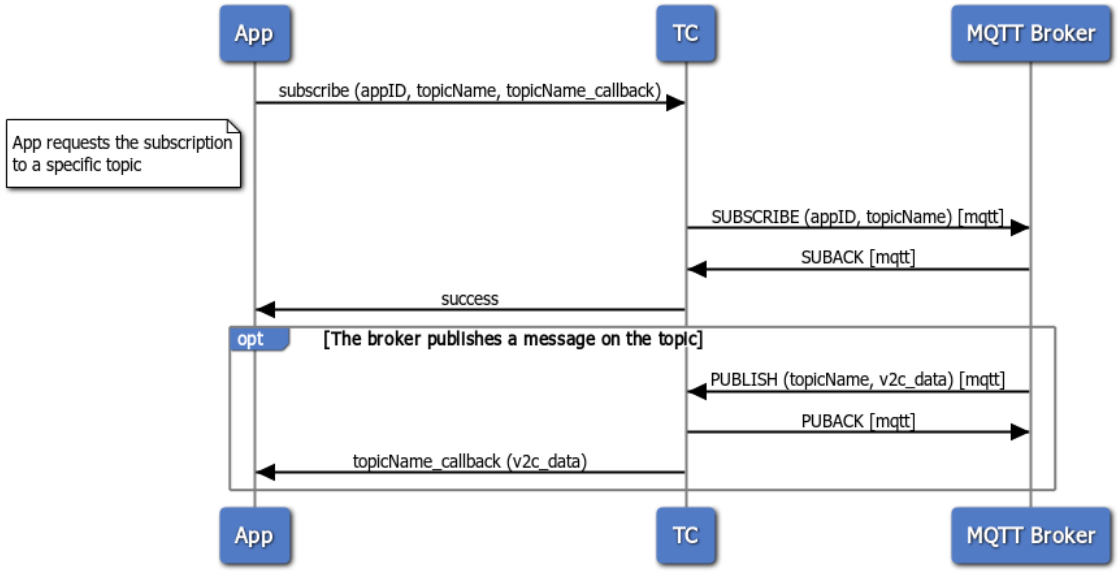

Figure 6.4: Sequence diagram of the proposed example.

6.2 Use cases for telematic box

Some possible use cases for the analyzed box flows can be the following:

1. As a call center operator, I want to be able to get speed, position, beltStatus and airbagStatus after a real accident, when the SOS feature is launched. It is crucial that cloud forwards the most updated data to provide the right information to the ambulance.[fig.6.5]

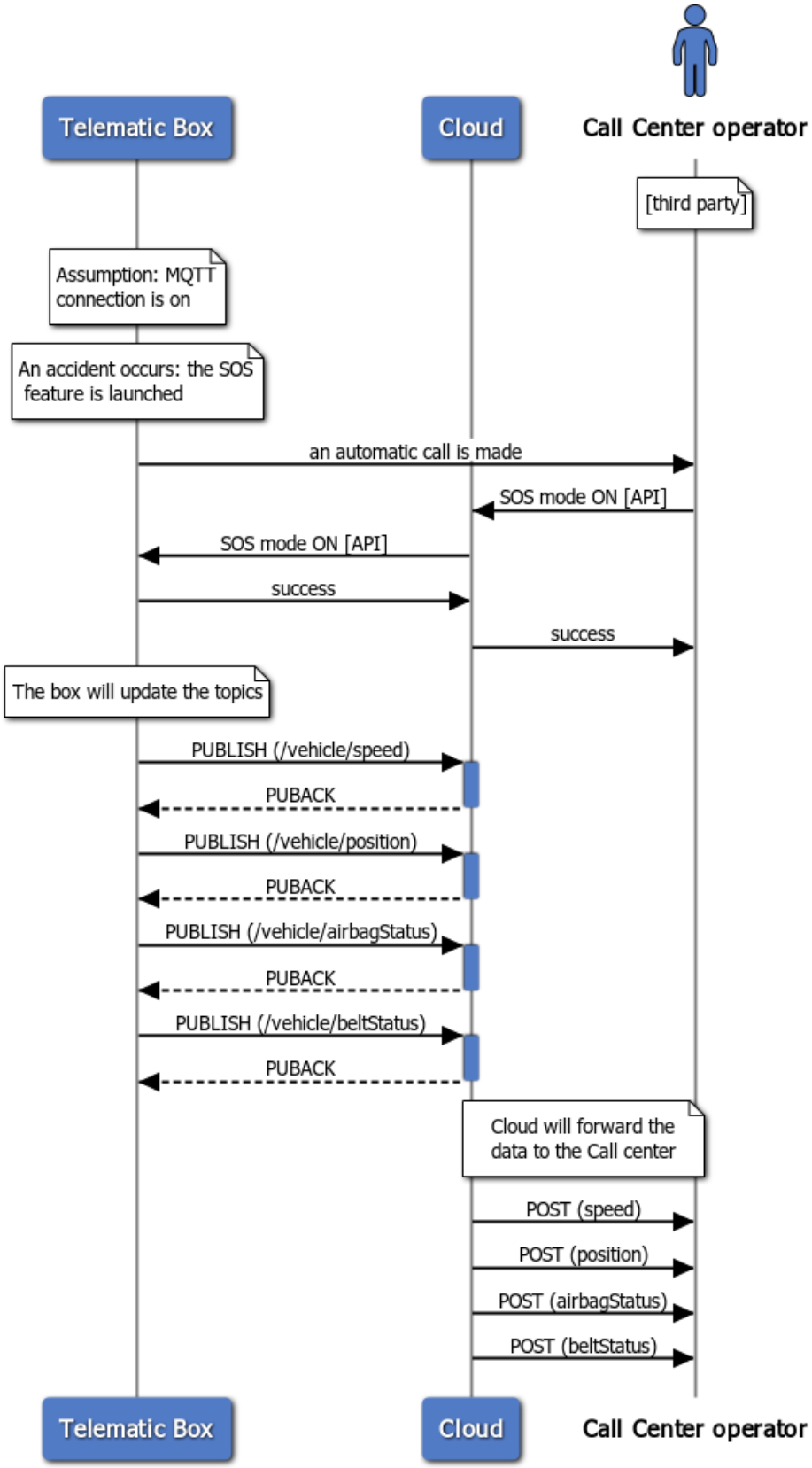

Figure 6.5: Sequence diagram of the first use case.

2. As a call center operator, I want to be able to retrieve the live vehicle position, when a stolen case is opened.[fig.6.6]

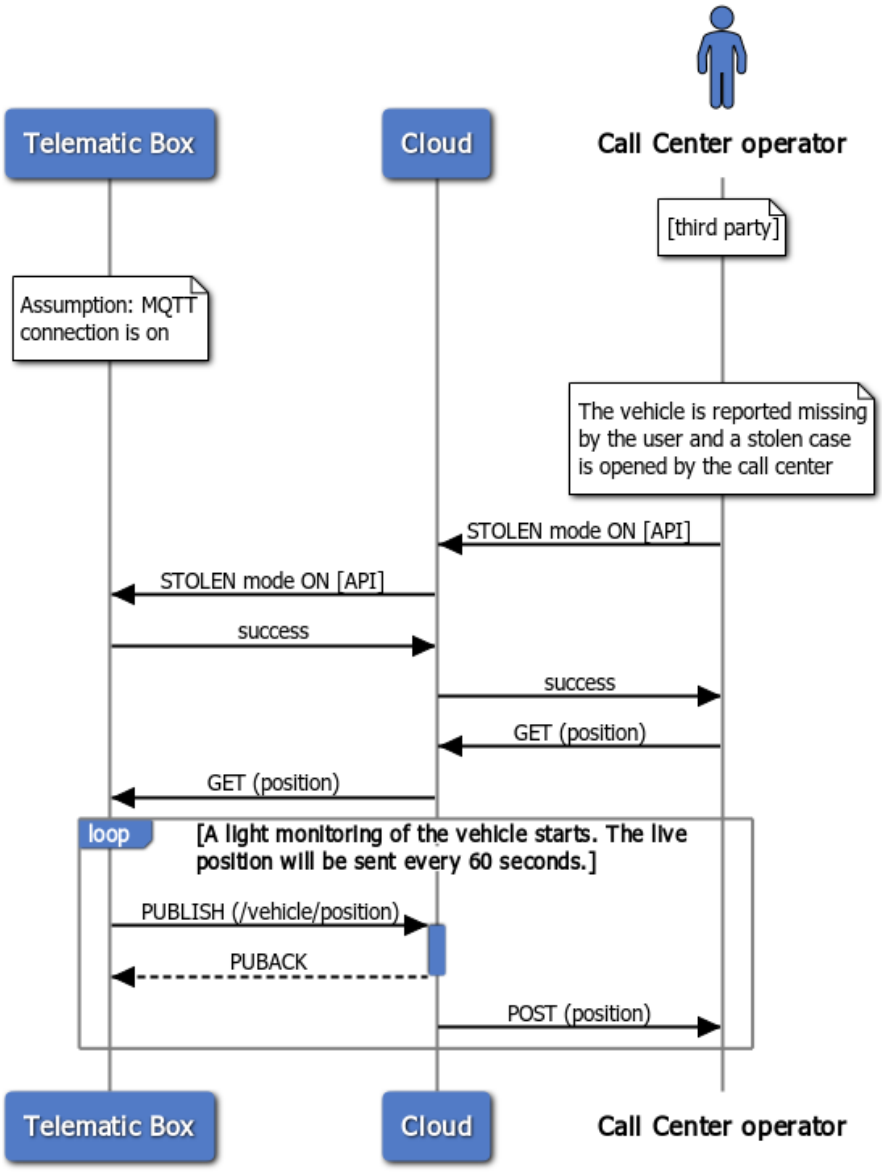

Figure 6.6: Sequence diagram of the second use case.

3. The custom level requires all the data necessary to create a VHR (Vehicle Health Report), to be distributed monthly to the customer via email, but if a value above the threshold is received, the custom level will evaluate on a case-by-case basis what to do and how urgently to contact the owner of the vehicle.[fig.6.7]

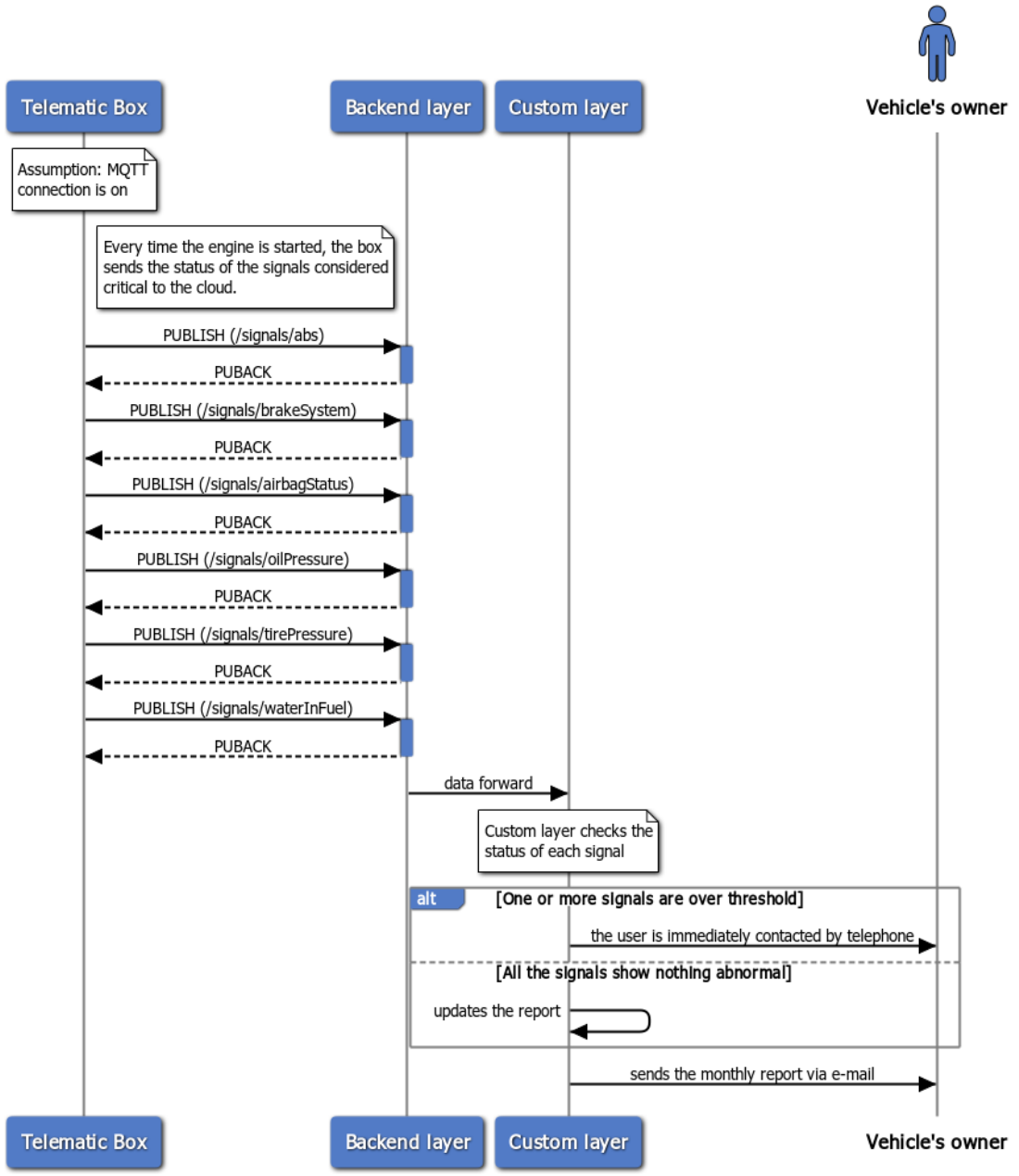

Figure 6.7: Sequence diagram of the third use case.

Note: For all the use cases described, this assumption applies: the box publishes with QoS equal to 1, because for these scenarios robust data delivery is expected, but the presence of duplicates can be tolerated. Furthermore, as already mentioned, it could be a good implementation choice to set the retain flag to true, so that the broker can store some information and immediately provide it to the client

without waiting for the box to publish an update.

6.3 The new role of mobile applications

Until a few years ago, there were already apps that provided information about the vehicle, but they were almost always sold as after-sales to be added, paid or not.
With the paradigm change linked to the obligation to insert a telematic box on the car and to bear the mandatory costs connected to it, the main OEMs have decided to take advantage of this, connecting it to particular control units to obtain useful information about the vehicle and send it to the application. More attention has therefore been paid to the mobile application, making it no longer an additional after-sales element, but a real component of the production chain of the connected vehicle.

6.3.1 Mobile app-cloud: MQTT flows

After a classic subscription procedure to the interested topics, MQTT is used on the mobile app to exchange information with the cloud [fig.6.8], in two different ways:

- passive mode;
- active mode.

Figure 6.8: Mobile Application - Cloud flows.

The first mode is the one used for those cases in which the user opens the app finds on the homepage static information listed and statistics (for example, the user sees how many kilometres he or she can travel before recharging). These are data that are not actively requested by the user. The second mode is the one that is used when the user *triggers* a specific request (for example, to retrieve the live position).

Passive mode

As already mentioned, being MQTT one of the protocols that is well suited to be used when offline, thanks to those mechanisms for which it then sends updates when the client comes back online, in specific scenarios it makes sense to start a sort of synchronization first and then trigger some reasoning on the application based on the context. This means that, in passive mode, MQTT could be used by the mobile app to receive information on data update status, even before the full package is actually sent.
To do this, the cloud component that communicates with the app, i.e. the backend, could therefore first send only a field that acts as a *watcher mechanism*, such as a timestamp.
This can be a possible choice motivated by the assumption that the MQTT channel requires high performance and there is no point in transmitting a large amount of data through it. Moreover, it may happen that data sent from the backend is not required and therefore may not be used, so sending only the timestamp will be only a few bytes, much less than all the json (for example).
In this way, after receiving the timestamp, the mobile app will check it and evaluate whether or not to call a specific REST API to request the sending of the data.
So, in this context, MQTT can be used to trigger a data update

watcher thanks to which the mobile app will not have to always request the data, but rather the backend will send an alert "there can be new data" and only then the app will check if it is appropriate to download the packet.
This is useful in scenarios such as those where the user parks the car in the evening, assuming that the next day he or she will find it in the same place: if the position does not change, it is useless to download it every time the user opens the application, but this still guarantees that the mobile app will be updated in real time. This "sync" process is an enabler and not a real feature, but using the timestamp to drive app updates can be an interesting use case and in fact MQTT topics are used a lot for updates on positions, vehicle status, speed warnings, battery percentage, etc.

Active mode

On the contrary with respect to what has just been presented, for some features the status update is always mandatory: this means that the mobile app will need to receive the complete data.
For these flows, the customer will trigger a certain event by tapping something on the mobile app: in these cases, the sending of a lot of data is justified by the fact that the user is actively using the app and this is not to be considered a waste of data.

6.4 Use cases for mobile application

Some possible use cases for mobile application flows can be the following:

1. As a private user, I want to see PHEV (Plug-in Hybrid Electric Vehicle) details on the battery percentage of my vehicle in

the homepage of the mobile app (*passive mode* for customer app).[fig.6.9]

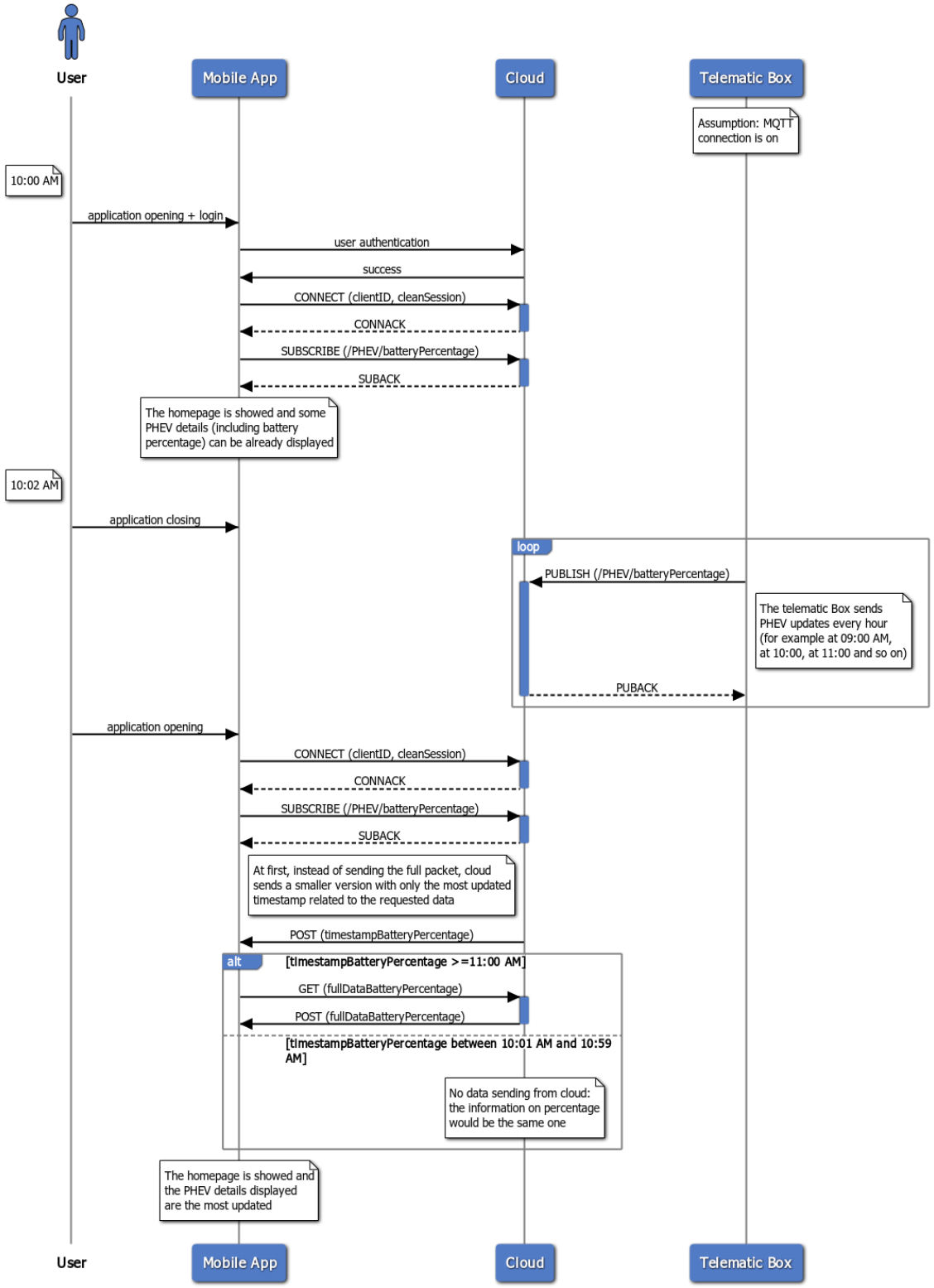

Figure 6.9: Sequence diagram of the first use case.

2. As a private user, during a charging, I want to see PHEV live statistics on current estimated time to fully charge my vehicle (*passive mode* for customer app).[fig.6.10]

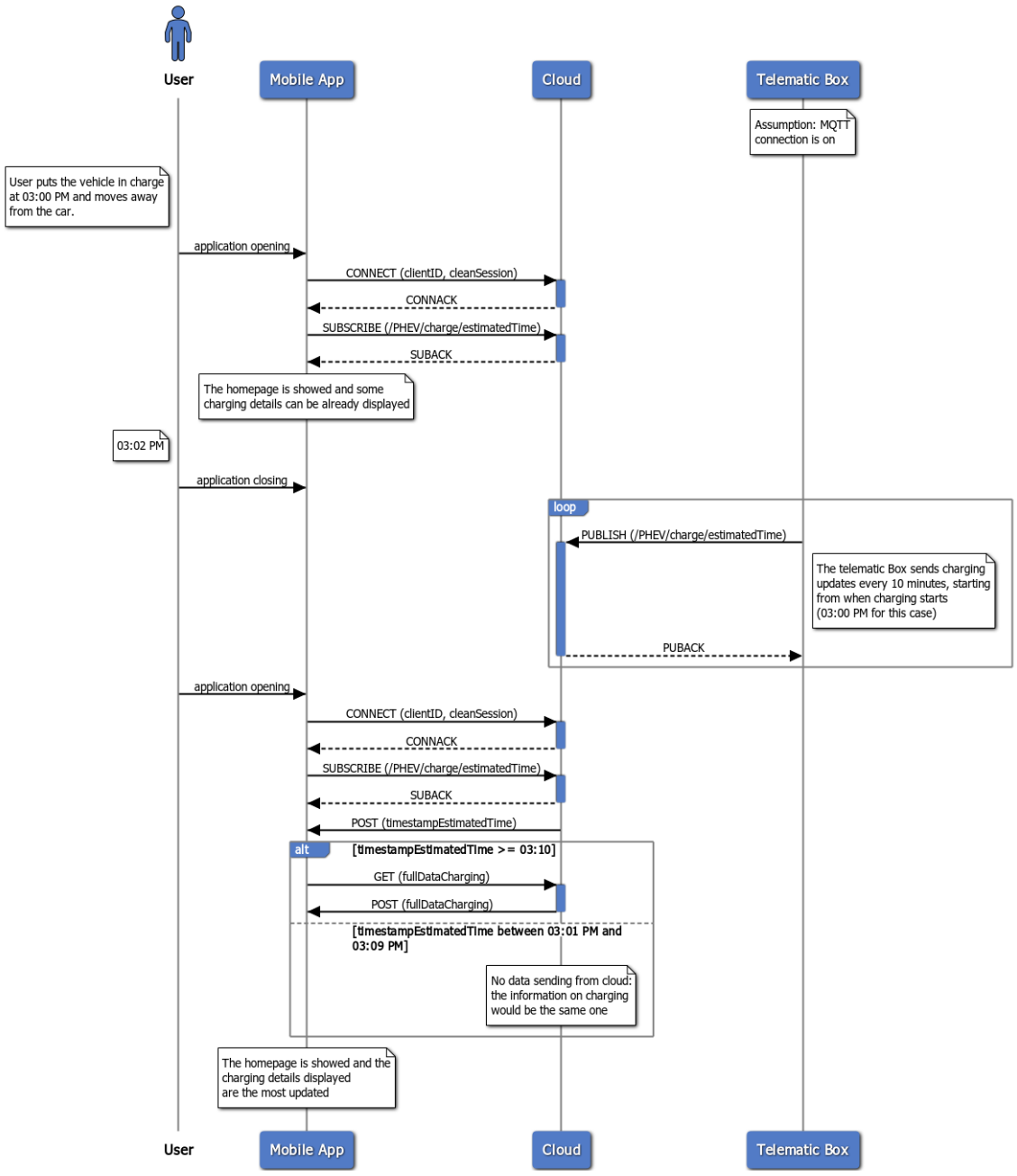

Figure 6.10: Sequence diagram of the second use case.

3. As a user of customer fleet mobile app, I want to launch a remote operation for door locking/unlocking without the need for a key (*active mode* for customer fleet app).[fig.6.11]

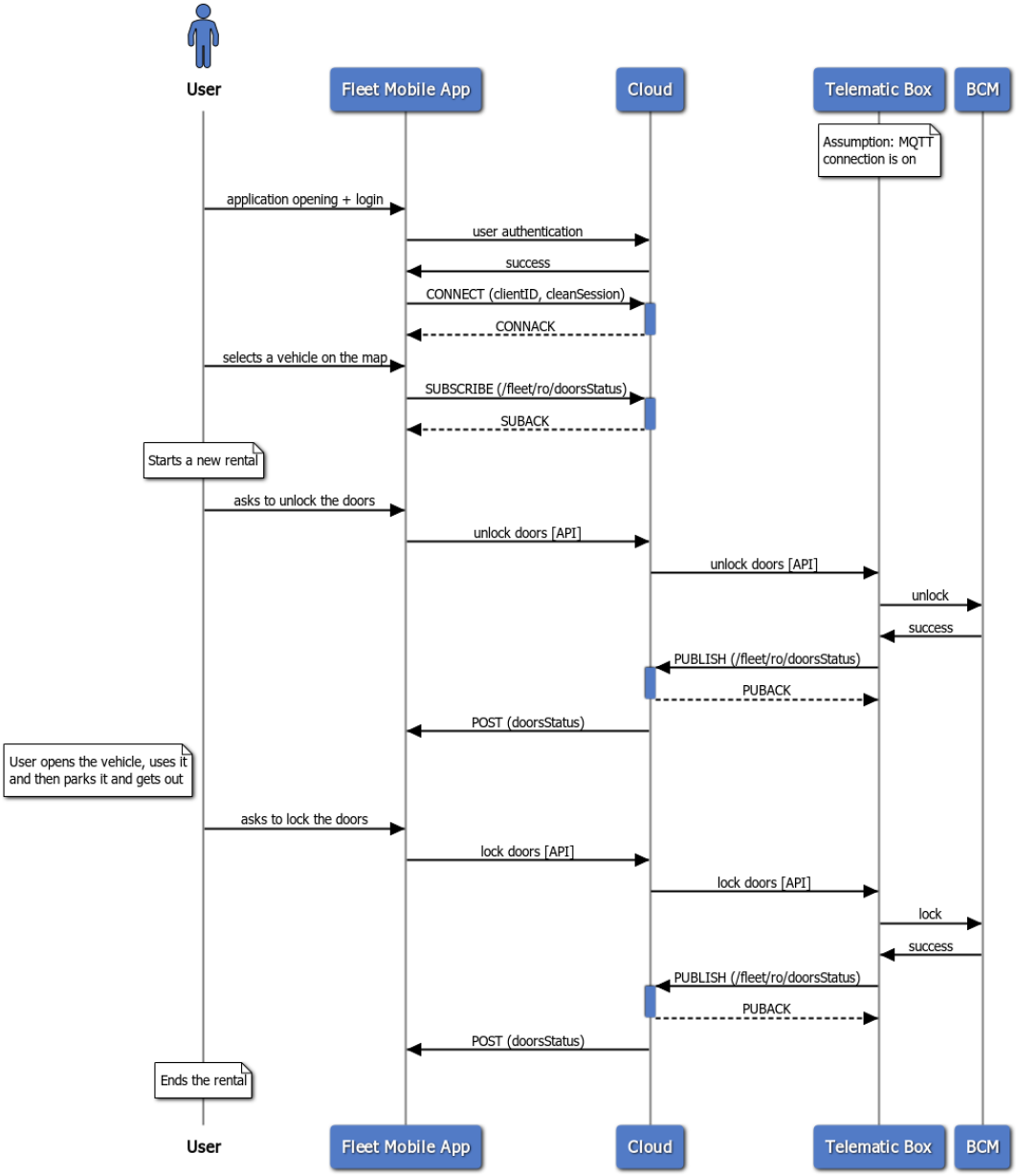

Figure 6.11: Sequence diagram of the third use case.

Two more possible examples of *remote operations* which have a very similar flow (with the only difference being applied to the customer app) and therefore not represented can be the following:

- the activation and flashing of the headlights in case the user forgets where he/she parked;
- the activation of a set of resistors placed on the windshield that allow it to be preheated and defrosted before the user

gets on board.

4. As a private user, I want to be notified if my vehicle has been stolen and if the call center operator in charge of the case manages to activate a remote crank of the engine, if the environment allows to stop the car safely (*passive mode* for customer app + MQTT flow between cloud and third party).[fig.6.12]

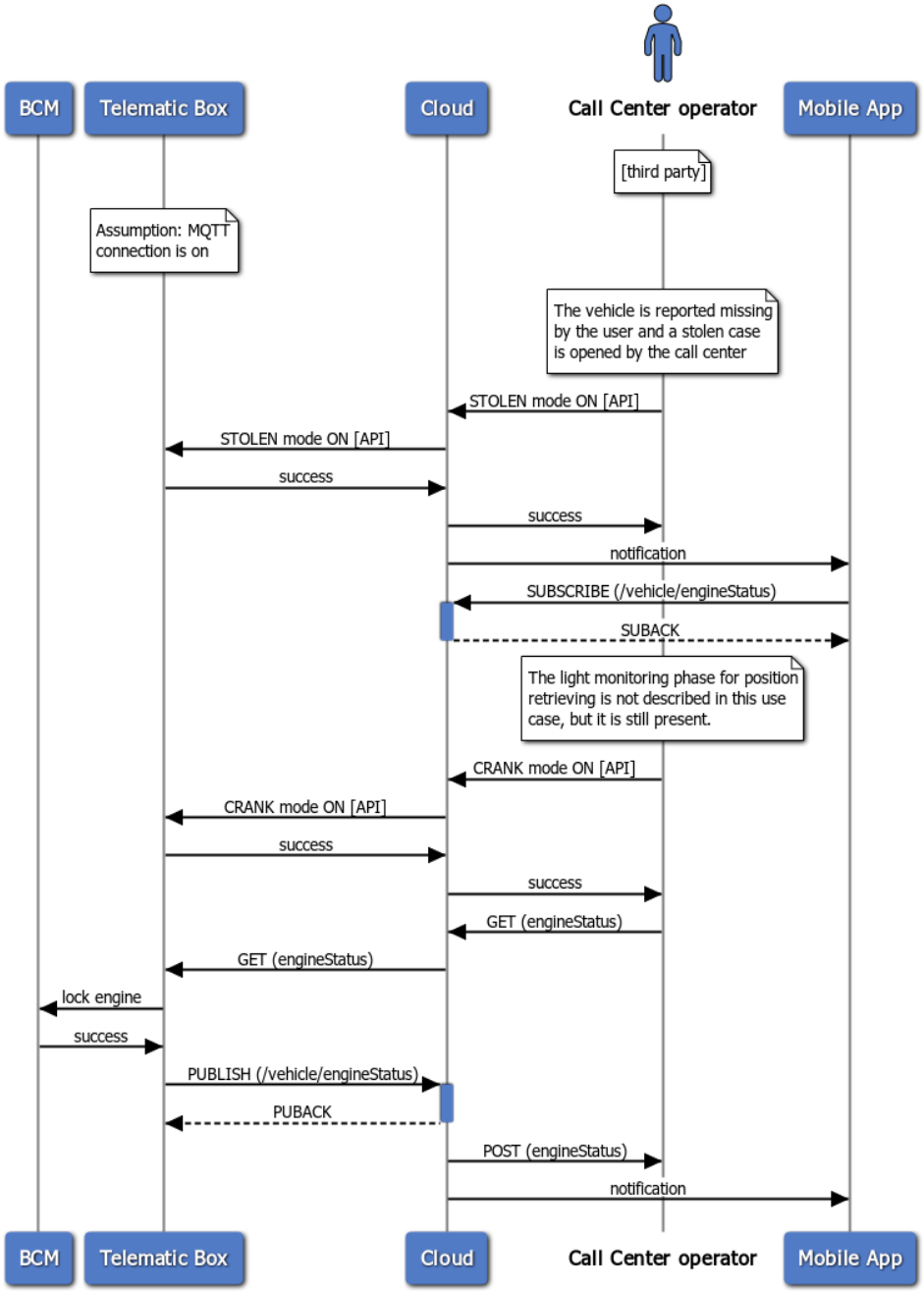

Figure 6.12: Sequence diagram of the fourth use case.

Note: As for the use cases of the box, also for those related to the mobile app it is assumed that the box publishes with QoS equal to 1, for the reasons already listed, and it is advisable to set the retain flag to true. Also, since a mobile app is a type of client that often may not be connected to the topic, for PHEV-related operations in the connect messages the CleanSession Flag could be set to 0, since this type of feature often has remote configurations for which it would be better for the broker to remember client subscriptions rather than create a new session each time a new request of connection is received.

6.5 Head unit-cloud flows

Since the Head Unit does not connect directly to the MQTT broker, because it connects to the telematic box that contains the modem and then it will be the latter that sends the message, it was chosen to not analyze in this work any use case related to the HU.

Chapter 7

Conclusions

The MQTT protocol, widely adopted in the automotive scenario, proves to have reached a certain degree of maturity. It fits well in automotive context for all the reasons reported in the course of this work and especially because the telematic boxes can not always be connected, but this constraint is well managed by the options provided by the protocol itself.

However, there are some issues for which there is certainly room for improvement. Above all, it must be ensured that only the vehicle owner has access to the data collected and therefore the world of connected vehicles opens up to a series of reflections on safety and privacy that must be regulated.

Furthermore, the numerous sensors inside the connected vehicle collect many information, but not all data are useful. The telemetry data streams must always be deeply analyzed and often combined with other data, in order to give manufacturers insights into the operational behavior and performance of the vehicle.

MQTT packets represent a significant slice within the data analysis traffic process and therefore require adequate tools: in this regard, the use of the "splunk" tool during this thesis work was very useful and allowed to carry out the necessary capturing logs, filtering and monitoring operations. This also demonstrates that the software market

believes in the relevance of the protocol and of the connected world in general, so much so as to create special tools to be sold to companies so that they can conduct predictive monitoring, also through the use of customizable dashboards.

7.1 Future developments

In general, the continuous investments aimed at improving the connected vehicle field constitute a great capability. As a matter of fact, not only OEMs will benefit from the advantages of the connected world widely discussed in this study, but also third-party companies could do so, because the parent company can share the information that the box collects and this is very interesting and can open to many other services that we do not even imagine (for example, a courier could make a delivery by leaving the package inside customer's vehicle thanks to the possibility of opening the hood with a mobile app).

The primary focus of this thesis work was a vehicle-cloud and app-cloud study conducted from the company's point of view, but as a future development the world of integrations with third parties could be explored. With the introduction and affirmation of electric vehicles, for example, integrations could be made on the mobile app towards companies that manage charging stations, or agreements could be made with partners such as insurance companies which, based on the guide style of the driver (evaluated thanks to the MQTT data collected and made available by the OEM), could implement *gamification* mechanisms to reward their customers or to attract new ones.

The same use cases described in chapter 6 could be applied to a different context, in which for example the telematic box speaks a different IoT protocol, such as LwM2M or CoAP, or uses the new version

of the same one (MQTT v5). A possible in-depth study could therefore be to compare a box implemented in the analyzed way with a box with another implementation in order to make a performance evaluation. Or it may be possible to conduct an analysis aimed at tuning some MQTT parameters, with monitoring aimed at understanding, for example, the impact of persistent sessions and retained messages.

Another next development could be focusing on the Head Unit and analyzing its integrations with systems such as Alexa, Google Assistant and Siri.

Bibliography

[1] Milica Matić, Marija Antić, Sandra Ivanović, and Ištvan Pap. Scheduling messages within mqtt shared subscription group in the clustered cloud architecture. In *2020 28th Telecommunications Forum (TELFOR)*, pages 1–4. IEEE, 2020.

[2] Biswajeeban Mishra and Attila Kertesz. The use of mqtt in m2m and iot systems: A survey. *IEEE Access*, 8:201071–201086, 2020.

[3] Tim Pulver. *Hands-On Internet of Things with MQTT: Build connected IoT devices with Arduino and MQ Telemetry Transport (MQTT)*. Packt Publishing Ltd, 2019.

[4] `https://www.oracle.com/it/internet-of-things/what-is-iot`.

[5] ISO Central Secretary. Information technology — message queuing telemetry transport (mqtt) v3.1.1. Technical Report ISO/IEC TR 20922:2016, International Organization for Standardization, 2016.

[6] `https://mqtt.org`.

[7] Stephen Cope. *MQTT for Complete Beginners: Learn The Basics of the MQTT Protocol*. 2020.

[8] Federico Maggi, Rainer Vosseler, and Davide Quarta. The fragility of industrial iot's data backbone. *Trend Micro Inc*, 2018.

[9] Cases Philippe Marshall Phil. Enabling the connected vehicle market to thrive.

[10] Deloitte LLP. Disruption in the automotive industry; enhancing the customer experience through connectivity, 2018.

[11] Riccardo Coppola and Maurizio Morisio. Connected car: Technologies, issues, future trends. *ACM Comput. Surv.*, 49(3):46:1–46:36, 2016.

[12] Randall Perrey and Mark Lycett. Service-oriented architecture. In *2003 Symposium on Applications and the Internet Workshops, 2003. Proceedings.*, pages 116–119. IEEE, 2003.

[13] Jian Wang, Yameng Shao, Yuming Ge, and Rundong Yu. A survey of vehicle to everything (v2x) testing. *Sensors*, 19(2):334, 2019.

[14] Canducci Alessi Maradei Castiello D'Antonio Abbiati, Coraretti. Digital ecosystems & composable solutions. https://www.eng.it/white-papers/digital-ecosystems-mobility.