People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

UNIVERSITY KASDI MERBAH OUARGLA

Faculty of New Technologies of Information and Communication

Department of computer science and Information Technology

Mémoire

With a view to obtaining the diploma of

MASTER Informatique

(Specialty: Network administration and security)

Presented by:

BEN GHEDIER CHAIMA

KOUIDRI AMAL

Title

Comparative Study of Communication Protocols in the IoT

A. **Encadreur : KHELILI Khalida Farida**
B.

Annie universitaire :2023/2024

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

وَأَنَ لـيْ َسَ لِلإِنسَانِ إِلاَّ مَا سَعَى ايـة 39

وَأَنَّ َ سَعْيَهُ سَوْفَ يرَ ُاي ايـة 40

صَدَقَـا لُله الـعظيَمُ

الاهداء

من قال أنا لها .. نالها

وأنا لها وإن أبت رغما عنها أتيت بها

لم تكن الرحلة قصيرة ولا ينبغي لها أن تكون، ولم يكن الحلم قريبا

ولا الطريق كان محفوفًا بالتسهيلات لكنني فعلتها وتلتها.

(آخِرُ دَعْوَاهُمْ أنِ َ الْحَمْدُ لِلّ ّ رَ بِ الْعالَمِ َينَ)

إلى من شرفني يحمل اسمه .. والدي العزيز)قويدر(حفضه الله إلى النور الذي أثار دربي والسراج الذي لا ينطفي نوره بقلبي أبدا من بذل الغالي والنفيس واستمديت منه قوتي واعتزازي بذاتي .

إلى نور عيني وضوء دربي ومهجة حياتي .. إلى التي ساندتني، ووقفت بجانبي، وقدمت لي الدعم المواصلة طريقي إلى التي وهبتني الحياة والأمل، واحتضنني قلبها قبل يدها وسهلت لي الشدائد بدعائها، والدتي)فاطمة (الحبيبة حفظها الله.

إلى ضلعي الثابت وأمان أيامي إلى ملهمي نجاحي إلى من شددت عضدي بهم فكانوا لي ينابيع أرتوي منها إلى خيرة أيامي وصفوتها إلى قرة عيني أخواتي "نسرين " " شريفة"" وفاء" شهد" اخواني " حمزة " ابوبكر" بلال "عبد القادر"حفضهم الله

إلى حبيبة قلبي بنات اختي "رباب" "رتاج" حفضهم الله

كل من كان عونا وسنداً في هذا الطريق .. للأصدقاء الأوفياء ورفقاء السنين وأصحاب الشدائد والأزمات إلى من أفاضني بمشاعره ونصائحه المخلصة صديقتي)نور.(

هديكم هذا الإنجاز وثمرة نجاحي الذي لطالما تمنيته ها أنا اليوم أتممت أول ثمراته بفضل من الله عز وجل، فالحمد لله على ما وهبني، وأن يعينني ويجعلني مباركة أينما كنت

خريجتكم امال

الإِهْدَاء

إلا ما عاش فينا قبل أن نعيش فيه، وعرفناه في دفاتر التضحيات إلى وطننا الثاني فلسطين، قبلتنا الأولى، ومسرى حبيبنا ونبينا الكريم. جمعنا الله في أقصاها فاتحين مهللين مكبرين، وليس ذلك على الله بعسير.

مثل كل البدايات إلا بتيسيره وما بلغنا النهاية إلا بتوفيقه وما حققنا المراد إلا بفضله فالحمد لله الذي وفقنا الإتمام هذه الخطوة في مسيرتنا التعليمية

"سنشد عضدك بأخيك "إلى سندي وقوتي أبي الثاني أخي "اسامة."

الى تلك التي جف ماء عينيها لاجلنا ، إلى من كان دعاءها سر نجاحي، وضوء أملي في ظلام اليأس إلى ملاذي، وركن الثابت الذي لا يميل إلى أمي العزيزة" منيرة العيفة" دمتي قوتي وذخري في هذه الحياة.

إلى من أحمل اسمه بكل فخر، إلى سيد قلبي إلى من فرش لي ورود النجاح وسار على أشواك الحياة ، إلى أبي "علي."

إلى وطني الذي أنا من دونهم في غربة إخوتي "اناس" "أيوب" "بسمة" "إسلام" "فاطمة الزهراء" الذين تقاسمت معهم مر الحياة وحلوها، فدمتم لي ضماد الروح وبلسم الجروح.

إلى أولئك الذين أضاءوا طريقي وقدموا لي الدعم خلال رحلتي الدراسية، إلى رفاقي في الطريق"سقاي عبد المنعم " "ليلى غربي" "محمد امين قوبي"

وختاما، نسأل الله أن يتقبل منا سعينا ويبارك لنا فيه ونحمده حمدا جليلا وتصلي وتسلم على شفيع الأمة تسليما.

خريجتكم شيماء

Résumé

The Internet of Things is defined as the creation of various objects connected to the Internet. This includes phones, refrigerators, traffic systems, smart homes, street lights, and even people themselves, as these objects can communicate with each other and perform various tasks without human intervention. The goal of this technology is to improve the individual's life, make it safer and more comfortable, and help save time and effort. IoT technologies find applications in many fields such as healthcare, agriculture, industry, selfdriving cars, homes, smart cities, etc.

Because IoT devices are small, limited in power, processing power, and battery life, and rely on wireless sensor networks (WSNs) that have low bandwidth and unreliable connections, many of these devices cannot guarantee efficient and acceptable communication.

To improve IoT communications, several protocols have been proposed at the application layer (MQTT, AMQP, CoAP, etc.), and due to this diversity of (non-standard) protocols, choosing an application protocol is not easy.

This work aim is to help users to choose a communication protocol, by the study of the two most commonly used protocols: CoAP and MQTT-sn. In this study we have evaluated the two parameters: Power consumption, and throughput using the COOJA simulator, in deferent scenarios so a user can discover the protocols performance.

Keywords: Internet of Things, CoAP, MQTT-SN, simulation, Cooja, performance, evaluation.

ملخص :

يعُرَّف إنترنت الأشياء بأنه إنشاء كائنات مختلفة متصلة بالإنترنت. وهذا يشمل الهواتف والثلاجات وأنظمة المرور والمنازل الذكية وأضواء الشوارع وحتى الأشخاص أنفسهم، حيث يمكن لهذه الأشياء التواصل مع بعضها البعض وأداء مهام مختلفة

دون تدخل بشري. الهدف من هذه التكنولوجيا هو تحسين حياة الفرد، وجعلها أكثر أماناً وراحة، والمساعدة في توفير الوقت والجهد. تجد تقنيات إنترنت الأشياء تطبيقات في العديد من المجالات مثل الرعاية الصحية والزراعة والصناعة والسيارات ذاتية القيادة والمنازل والمدن الذكية وما إلى ذلك.نظرًا لأن أجهزة إنترنت الأشياء صغيرة ومحدودة في الطاقة وقوة المعالجة وعمر البطارية وتعتمد على شبكات الاستشعار اللاسلكية (WSNs) التي تتمتع بنطاق ترددي منخفض واتصالات غير موثوقة، لا يمكن للعديد من هذه الأجهزة ضمان الاتصال الفعال والمقبول.

لتحسين اتصالات إنترنت الأشياء، تم اقتراح العديد من البروتوكولات في طبقة التطبيق MQTT) و AMQP و CoAP وما إلى ذلك(، وبسبب هذا التنوع في البروتوكولات)غير القياسية(، فإن اختيار بروتوكول التطبيق ليس بالأمر السهل . يهدف هذا العمل إلى مساعدة المستخدمين على اختيار بروتوكول الاتصال، من خلال دراسة البروتوكولين الأكثر استخدامًا : CoAPو .MQTT-sn في هذه الدراسة قمنا بتقييم المعلمتين: استهلاك الطاقة، والإنتاجية باستخدام محاكيCOOJA ، في سيناريوهات مرجعية حتى يتمكن المستخدم من اكتشاف أداء البروتوكولات. الكلمات المفتاحية : إنترنت الأشياء ، MQTT-SN،CoAP، المحاكاة ،Cooja، الأداء، التقييم.

Table of contents

List of Figures

List of Table

List of abbreviations

AMQP: Advanced Message Queuing Protocol

CoAP: Constrained Application Protocol

DTLS: Datagram Transport Layer

DDS: Data Distribution Service

HTTP: Hypertext Transfer Protocol

IEEE: Institute of Electrical and Electronics Engineers

IHM: Interfaces Home-machine

IOT: Internet of Things

IPv6: Internet Protocol version 6

ISO: International Organization for Standardization

MQTT: Message Queuing Telemetry Transport
MQTT-SN: Message Queuing Telemetry Transport Sensor Network

M2M: Machine to Machine

NAT: Network Adresse Translation

QoS: Quality of Service

TCP: Transmission Control Protocol

UDP: User Datagram Protocol

WSN: Wireless sensor network

XMPP: Extensible Messaging and Presence Protocol

Introduction Général

Introduction

The development of the internet and computer networks has led to the creation of the Internet of Things (IoT), also known as the Internet of Objects. Thanks to this development, almost everything has become connected, enabling humans to access the internet from an increasing number of devices. These devices possess interconnected computer systems through networks that allow them to exchange data. Various types of technologies are required for the interact with each other, such as IoT protocols.

As of now, there is no standardization for IoT protocols, and users are unsure which protocol to choose because each protocol has its own characteristics and performances. For this reason, we have decided to study two of the most well-known IoT protocols, the MQTT protocol standardized by OASIS and the CoAP protocol standardized by the IETF.

To understand and study the MQTT-sn and CoAP protocols, we simulated them using the COOJA simulator in different scenarios to learn and analyze each performances in order to compare them. We choose to evaluate the two metrics : energy consumption and throughputs in four scenarios changing and increasing the number of servers and the clients in the first one for the two protocols, and changing and increasing the data transmitting interval for the two protocols too in the second one. The results of simulation where presented in curves and chart columns.

At the end of simulations, we have analyses for the results in order to explain them and made an experimental comptonization based on simulation between the two communication protocols CoAP and MQTT-sn , and giving advices to help users to make communication protocol choice for an IoT application.

Our work is comprised of a general introduction and four chapters:

Chapter 1: We discuss IoT, definition, functioning, and its various characteristics.

Chapter 2: This chapter deals with IoT communication protocols and their characteristics.

Chapter 3: This chapter presents the MQTT and CoAP protocols, explains how they work, and gives a theoretical corporatization between them.

Chapter 4: Is dedicated to presenting the simulation of the two protocols, MQTT-sn and CoAP.

Chapter 5: The results of the simulations is giving in this chapter, ending by analytical resume that have been conducted to evaluate and compare the MQTT-sn and CoAP protocols.

I. Chapitre 1 Internet of Things

I.1 Introduction

The Internet of Things (IoT) constitutes a network comprising physical objects such as devices, tools, vehicles, buildings, and various other elements embedded with electronics, circuits, software, sensors, and network connectivity, facilitating the collection and dissemination of data.[7] These interconnected objects within the Internet of Things operate autonomously, often without human intervention, and have witnessed significant advancements across various sectors including healthcare, transportation, and automotive industries. Notably, there have been notable strides in integrating living organisms with Internet-connected sensors. The development of IoT encompasses infrastructure, communication frameworks, interfaces, protocols, and standards.

This chapter delineates key aspects of the Internet of Things, beginning with elucidations of its definitions and applications across diverse fields. Subsequently, it delves into the architecture of the Internet of Things, highlighting prevalent protocols, particularly focusing on application protocols. Finally, it addresses several challenges pertinent to the Internet of Things.

Some History:

The term 'Internet of Things' was coined in 1999 by Kevin Ashton, a British researcher at MIT. He launched an initiative to promote open connectivity between objects. All connected objects use RFID technology, which allows them to be identified remotely. With the arrival of the new IPv6 protocol, sectors such as aerospace are beginning to move away from RFID.

I.2 Internet of things

I.2.1 The Internet of Things Definition

Définition 1

The Internet of Things (IoT) refers to a set of technologies that enable the connection of physical objects to the Internet. These objects are capable of identifying, collecting, storing, processing, and transferring data in the real world.[01]

Définition 2

It can be defined as "an object endowed with virtual identity and personality, operating in intelligent spaces and utilizing smart interfaces to connect and communicate within a variety of usage environments." [02]

This definition emphasizes the intelligent aspect of connected objects. They are capable of collecting data, processing it, and communicating with each other. They can also interact with humans intelligently, for example, by responding to voice commands or offering recommendations.

On the other hand, IoT can also be considered as a ubiquitous network. It allows people to connect with each other anywhere, anytime, through any object. This definition emphasizes the

ubiquitous aspect of IoT. Connected objects are all around us, and they can be used to communicate with each other, even if they are distant. [03]

Figure 1:Internet of Things [41]

I.3 Applications of the Internet of Things

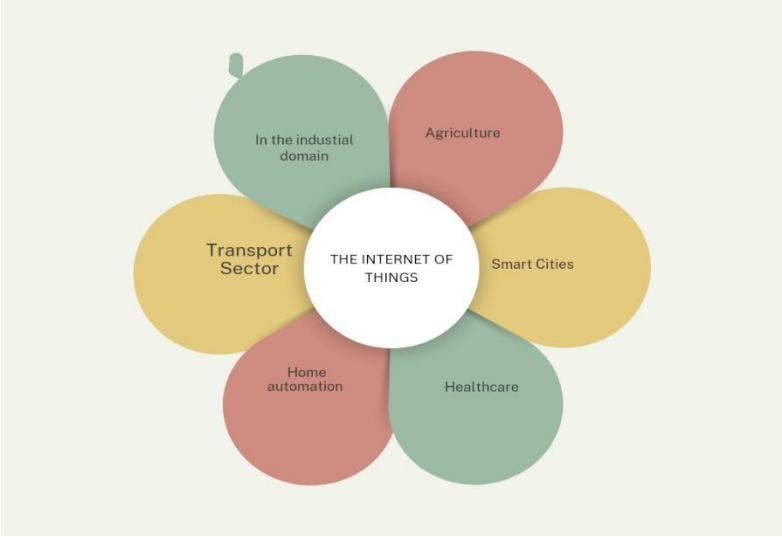


Figure 2:IoT application area [41].

The Internet of Things (IoT) is utilized in a variety of fields, including agriculture, healthcare, home automation, and more.

- **Home automation:** It is a set of technologies that enable homes to become intelligent. Thanks to the Internet of Things, household appliances can communicate with each other and

be controlled remotely. This allows homeowners to control their comfort and safety, and reduce their energy consumption. [05].

•**Agriculture:** The Internet of Things (IoT) can be utilized to monitor crop environments through interconnected sensor networks [7]. This leads to positive outcomes, including improvements in irrigation water management, efficient use of inputs, and better planning of agricultural activities. Additionally, these networks can be leveraged to mitigate damages and disasters while enhancing overall environmental quality.

- **Smart Cities:** Are cities that use digital technologies to improve the lives of their inhabitants and the efficiency of their administration [7]. By using innovative services, it is possible to optimize the use of the city's physical infrastructure (such as roads, electricity networks, etc.), which improves the quality of life of residents.

•**Healthcare:** The Internet of Things (IoT) has the potential to revolutionize the healthcare industry by allowing patients to monitor their vital signs remotely. Connected medical sensors can collect data on body temperature, blood pressure, and respiratory activity, which can then be transmitted to clinics or healthcare professionals. This information can be used to monitor patient health, detect potential problems, and provide more personalized care [05]. To monitor and provide solutions to people with reduced mobility in healthcare, wearable devices (such as accelerometers, gyroscopes, etc.) or fixed sensors are used to monitor their activities in their living environment.

•**Transport sector**: The Internet of Things has created connected, intelligent, and autonomous cars. These cars can save lives and improve the environment by reducing road traffic.

•**In the industrial domain**, the Internet of Things (IoT) enables the tracking of products throughout their life cycle, from production to distribution to supply. This comprehensive traceability enables factories to enhance their operations, improve production efficiency, and ensure the safety of their employees. [4]

I.4 How IoT works

I.4.1 IoT components

An IoT system is made up of many elements, including objects, networks, data, information, and applications. These elements interact together to allow connected objects to function. [6]

- **Objects (sensors):** Sensors are devices that measure physical quantities, such as temperature, brightness, or movement, and transform them into digital data. There are many types of sensors, and they are used in a wide variety of applications. Connected objects often use sensors to collect data about their environment. [08]

- **The network:** The sensors are equipped with wireless devices to communicate with each other. However, this is not enough to make these sensors accessible in an interoperable, transparent, and simplified manner. To achieve this, the sensors must be organized into a

network. Sensor networks are composed of very small devices with wireless transmission capabilities. [07]

•**Data:** In an IoT project, data is the most important resource. It is collected from connected objects such as sensors, cameras, or industrial machines. This raw data must be stored, archived, and structured in databases to be utilized effectively. A properly structured database enhances the performance of IoT services. [06]

- **Information**: stored, archived, and saved in databases, raw data is collected by connected objects. This data is then processed, correlated, and analyzed to derive insights. This information must be stored, archived, and saved in databases for future use. [06]

- **Operating applications:** Operating applications are human-machine interfaces (HMI) that allow users to view data in the form of dashboards. [06]

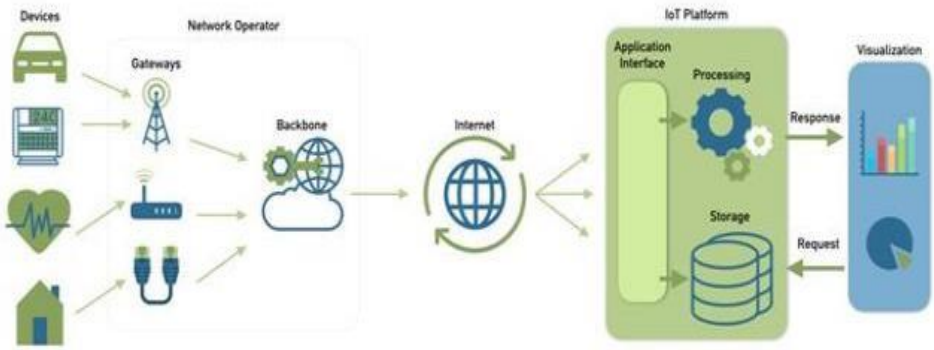


Figure 3 : How the Internet of Things works [10]

I.4.2 IoT technologies

The Internet of Things (IoT) allows smart objects to connect to each other over the Internet. For this to work properly, several technologies are required. Although there are many, we will focus on four: machine-to-machine communication (M2M), radio frequency identification (RFID), wireless sensor networks (WSN), and Bluetooth.

- **M2M**: IoT is a technology that enables everyday objects to connect to the Internet and communicate with each other. These objects can be equipped with sensors to collect data about their environment, and they can use this data to take actions or interact with other objects. [11]

●**RFID:** is a technology that automatically recognizes objects or people using radio waves. It is based on two main characteristics: storage and remote retrieval of information.

- **WSN:** A cooperative network is one in which nodes work together to achieve a common goal. Each node in the network has a set of characteristics that allow it to contribute to the common goal. These features may include processing power, different types of memory, RF transceivers, power supplies, and various sensors and actuators. [10]

- **Bluetooth:** is a short-range wireless communication technology (around 10 meters) which allows large messages to be sent in large quantities. However, it cannot work alone and requires other technology to transfer and store data. Zigbee is also a means of high-speed communication, as it uses the 2.4 GHz frequency band, just like Wi-Fi. [06]

I.5 IoT architecture

The rapid development of IoT raises the question of the usefulness of a reference architecture. Could such an architecture help standardize system design and foster interoperability and communication between different IoT ecosystems? [12] **A. The three-layer architecture:**

This architecture consists of:

application layer is responsible for providing specific application services to users. It defines various applications in which the Internet of Things can be deployed, such as smart homes, smart cities, and smart healthcare. [35]

The network layer (abstract layer) is responsible for communication between connected objects, network devices, and servers. It also enables the transmission and processing of data collected by sensors. [35]

The perception layer (things layer): The perception layer, or physical layer, is the layer of the Internet of Things that allows objects to collect data about their environment. This data can be physical measurements, such as temperature, speed, or humidity, or information about other smart objects in the environment. [35] **B. The five-layer architecture:**

The architecture in question is made up of five layers:

The perception and application layers of the Internet of Things (IoT) play the same role as the physical and application layers of the three-layer architecture**.**

The transport layer is responsible for transmitting data from the sensor to the processing layer. It uses networks such as wireless, 3G, LAN, Bluetooth, RFID, and NFC to connect the different layers of the IoT architecture. [35]

The processing layer is responsible for storing, analyzing and processing the data coming from the perception layer. It can also provide services to lower layers, such as data management, computation and decision-making. It uses many technologies, such as databases, cloud computing and big data processing. [35]

The business layer of the Internet of Things (IoT) is responsible for managing the entire system, including applications, business models, and user privacy issues. [35]

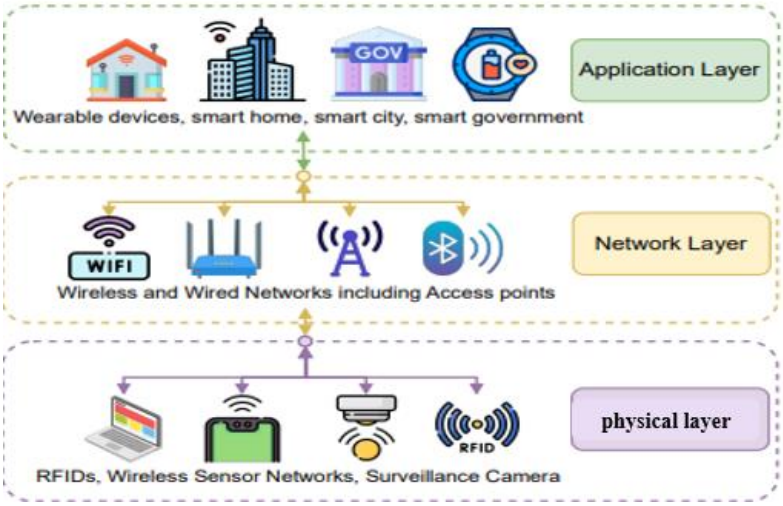


Figure 4 : IoT architecture [22]

I.6 The protocols of l'IoT

According to the three-tier architecture of the Internet of Things, the physical layer facilitates packet transmission across network segments. Meanwhile, the transport layer establishes communication channels for data transmission, which are then utilized by the application layer. The application layer holds particular significance as it encompasses the protocols utilized by users for service provision or data exchange..

application layer	**HTTP ,CoAP ,EBHTTP, LTP ,SNMP ,IPFIX, DNS ,NTP ,SSH DLMS ,COSEM ,DNP, MODBUS,MQTT,MQTT-SN**
Network layer	**IPV6 /IPV4 ,RPL ,TCP/UDP ,IUP ,SLIP ,6LOWPAN**
physical layer	**IEEE802.11 Series,802.15 Series,802.3,802,16,WirelessFART ,Z-WAVE ,UWB,IrDA, PLC,LonWorks, KNX.**

Table I-1 : Protocols of IOT layers Protocols

of application layer:

In an IoT setting, the application layer facilitates data exchange and communication among devices. Application layer protocols serve as the messaging frameworks utilized by devices to transmit data via the Internet. Presently, notable IoT application layer protocols include CoAP, XMPP, MQTT, DDS, AMQP, REST, WebSocket, and JMS.

I.7 Characteristics of an IOT application

The Internet of Things (IoT) draws its success from several key characteristics. Each feature offers a set of functionalities that contribute to the growth of IoT. Among the main characteristics of IoT, we can cite:

The connection: it is connected to IoT. It allows objects to be connected to the network at home and with other systems, senders and receivers, and access to the device. This collaboration is essential for the IoT function and for the realization of its simpler projects. [36]

Energy efficiency, quality, and reliability: are essential characteristics of IoT devices. These devices are frequently employed in extreme conditions, such as harsh weather environments, remote locations, or hard-to-access areas (for instance, deep within mines). To ensure their proper functioning under such conditions, it is crucial to manufacture them with high-quality materials, design them reliably, and optimize them for minimal energy consumption**.** [33]

The intelligence of IoT systems:

One of the main attractions of IoT is the intelligence demonstrated by its systems. Through a combination of algorithms and computing power, these systems are capable of:

• Analyze data collected by sensors to identify environmental changes.

• Make decisions based on these changes.

• Perform automatic actions to respond to detected situations. [36]

Security: is a crucial element for IoT adoption. Without adequate protection against cyberattacks and intrusions, users will not be inclined to use these systems. The sensitive nature of personal data processed by IoT requires the implementation of rigorous security measures. Although progress has been made in securing IoT systems, it is important to continue investing in this area to ensure an adequate level of protection. [36]

Sensors:
Central element of the Internet of Things :
Sensors play a crucial role in devices and systems connected to the Internet of Things (IoT). Their main function is to monitor, track and measure the activity and interactions of the device in question. Then, they transmit this collected information to the Cloud for further analysis and processing. [33]

Profitability: is a crucial element for the success of IoT. For connected objects to be effective, it is necessary to deploy them on a large scale. This implies that their production cost must be affordable. Take the example of a sensor affixed to food products to monitor the expiration date. To be effective, this sensor must be integrated into each product, which involves an additional cost. It is therefore crucial to find solutions to reduce this cost and guarantee the profitability of the entire system. [33]

I.8 Conclusion

The Internet of Things (IoT) has revolutionized the way we live and interact with the world around us. Thanks to the evolution of the Internet, intelligent objects can now communicate with each other and with their environment.

This chapter explores the fundamental concepts of IoT and its impact on various domains. We discussed the different application areas of IoT, as well as its architecture and operation.

II. Chapter 2: Communication in IoT

II.1 Introduction

Connectivity within the Internet of Things relies on a variety of protocols, each suited to different levels of the network stack and various system architectures. This chapter will explore communication in IoT in detail, highlighting the specific protocols used at the application layer.

II.2 The application layer

Application layer protocols are used to exchange data between programs running on source and destination hosts. There are application layer protocols that enable communication in IoT (XMPP, MQTT, CoAP, WebSocket, DDS, AMQP).

II.2.1 Application layer protocols

This passage explains that application protocols define the rules of communication between two computer applications. They use transport protocols such as TCP (Transmission Control Protocol) or UDP (User Datagram Protocol) to establish connections and exchange data according to the rules of the specific application protocol. Then he mentions that the most

commonly used application protocols will be listed.

A. XMPP (Extensible Messaging and Presence Protocol)

is a set of open standard protocols from the Internet Engineering Task Force (IETF) for instant messaging, and more generally a decentralized data exchange architecture? XMPP is also a near-real-time collaboration and multimedia exchange system through its Jingle extension, of which voice over IP network (Internet telephony), videoconferencing and file exchange are examples of applications.

XMPP is made up of a TCP/IP protocol using a client-server architecture allowing decentralized exchanges of instant messages or not, between clients, in Extensible Markup Language (XML) format. XMPP is under constant and open development within the IETF. [40]

- **Operation:**

"XMPP supports request/response and publish/subscribe models, enabling bidirectional and multidirectional communications. Its decentralized architecture ensures high scalability, and its numerous extensions allow it to operate without dedicated infrastructure." [41]

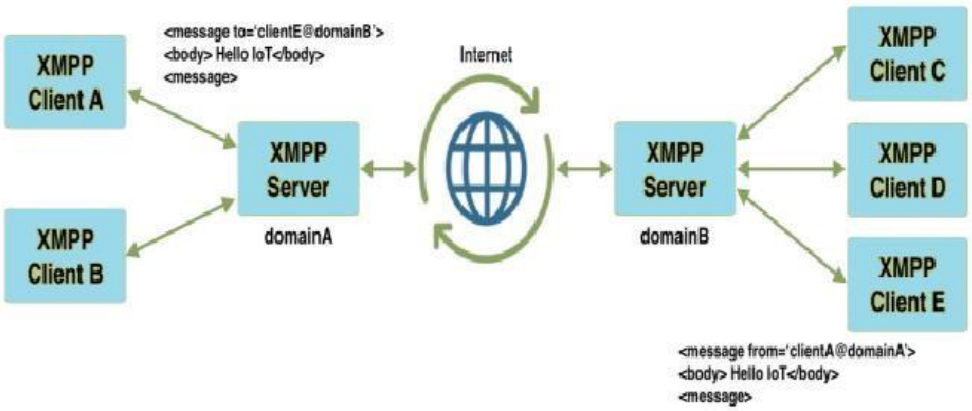


Figure 5: Operation protocol XMPP. [41]

B. MQTT (Message Queuing Telemetry Transport)

MQTT is a lightweight communications protocol specifically designed for IoT and M2M applications. It is ideal for remote environments or applications with limited bandwidth. MQTT uses a connection-oriented publish/subscribe architecture, in which MQTT applications can either publish (transmit) or subscribe (receive) topics, and an MQTT broker transmits the information from the publishing client to the subscribing client. [40]

- **Operation:**

A client, called publisher, first establishes a "publication" type connection with the MQTT server, called broker. Then, the publisher forwards the messages to the broker on a specific channel, called topic. [41]

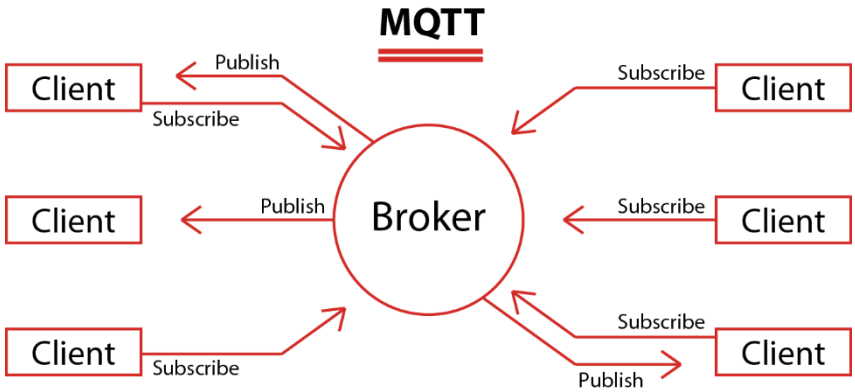


Figure 6 : Operation protocol MQTT. [41]

C. CoAP (Constrained Application Protocol)

Constrained Application Protocol (CoAP) is designed for low-power, lossy networks, also known as "constrained" networks. CoAP is typically combined with User Datagram Protocol (UDP), making it very efficient, making it attractive for IoT applications where battery

conservation is important. For example, it is often used in smart meter communications. CoAP can also use TCP or SMS as a transport mechanism. [40]

● **Operation:**

To transmit data via CoAP, a client sends a request containing several elements: the message type, the message identifier (mid), and an action (GET, POST, PUT or DELETE). Then, the server responds with a response that also includes the type of the message, the message identifier (mid), and a response code indicating the status of the request, optionally followed by additional data. [41]

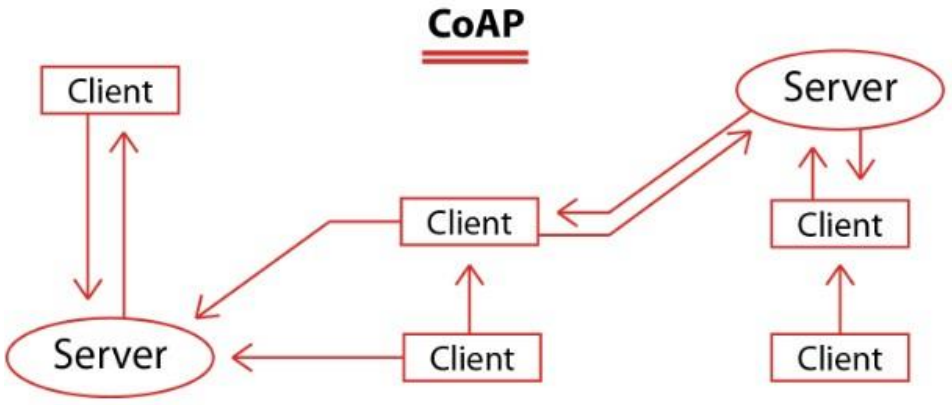


Figure 7: Operation protocol CoAP . [41]

d. AMQP (Advanced Message Queuing Protocol):

AMQP is an open-source protocol for Message-Oriented Middleware (MOM). It is designed to facilitate communications between systems, devices, and applications from multiple vendors and was not directly designed for IoT. [40]

● **Operation:**

The operation of the AMQP protocol is based on the same principle as that of MQTT, however the notion of publisher/subscriber is replaced by that of producer/consumer. In addition, thanks to an internal mechanism denoted “exchange”, AMQP makes it possible to route a message from a producer to several topics. Routing criteria can be done in several ways; inspection of content, header, routing keys, etc. Thus, the same message can be consumed by different consumers via several topics. [41]

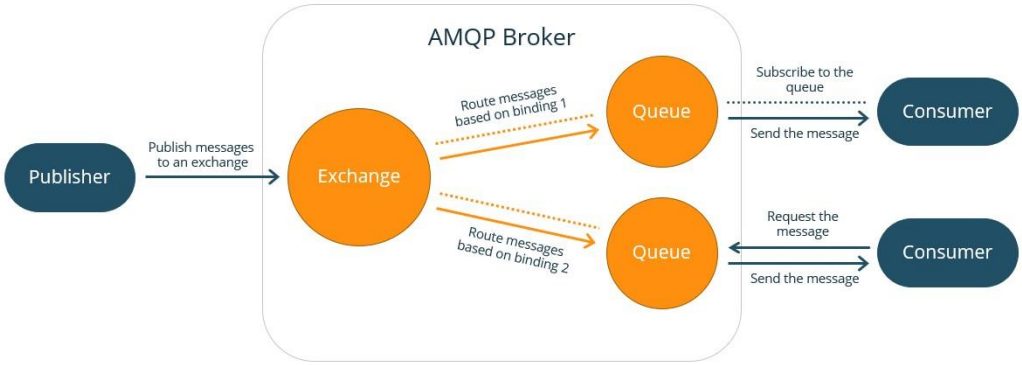


Figure 8: Operation protocol AMQP. [41] E.

WebSocket:

WebSocket is a two-way communications protocol designed to quickly send large amounts of data in web applications. A WebSocket establishes a connection between the client and the server and therefore after the initial establishment of the connection: each message has only a small overhead. Devices and servers can simultaneously transmit and receive data in real time, making this protocol best suited for IoT applications where low latency is essential, communications are frequent, and data consumption is less important. [40]

- **Operation:**

The WebSocket protocol makes it easy to establish a full-duplex communications channel over a single TCP connection between a client and a server. The three main phases of the life of the canal are as follows:

1. The connection phase, also known as "Handshake", is initiated by the client.
2. The bidirectional message exchange phase, where data can be transmitted simultaneously in both directions between the client and the server.
3. The channel closure phase, which can be initiated by either party to end the connection.

In short, the WebSocket protocol allows efficient, two-way communication between the client and server using a single TCP connection. [41]

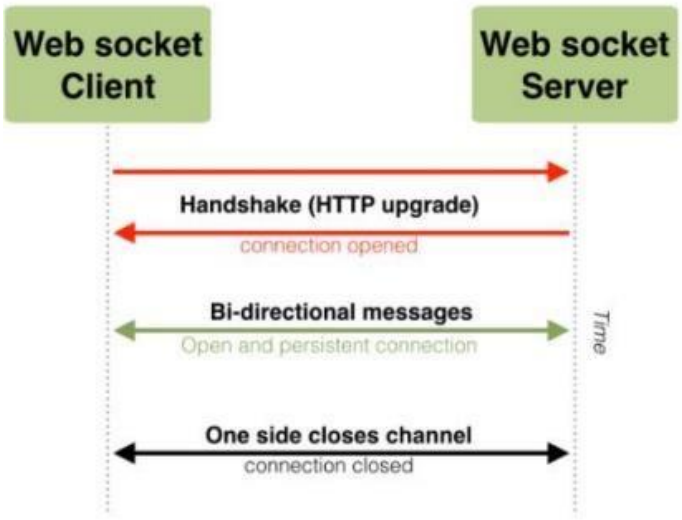


Figure 9 : Operation protocol WebSocket [41]

E. DDS (Data Distribution Service)

The Data Distribution Service protocol is a real-time, interoperable communications protocol designed for solutions that require significant coordination, reliable transmissions, and distributed processing between the devices themselves. Instead of sending data to a central hub or broker, data can be directly exchanged between peers, making it more robust and efficient. DDS uses a publish/subscribe mechanism in which devices subscribe to a topic and devices sending to the topic then use multicast to distribute the information to subscribers. DDS can use TCP and UDP as transmission protocol. [40]

- **Operation:**

The operation of the AMQP protocol is based on the same principle as that of MQTT, however the notion of publisher/subscriber is replaced by that of producer/consumer. In addition, thanks to an internal mechanism denoted “exchange”, AMQP allows you to route a message from a producer to several topics. Criteriarouting can be done in several ways; inspection of content, header, Routing keys, etc. Thus, different consumers via several topics can consume the same message. [41]

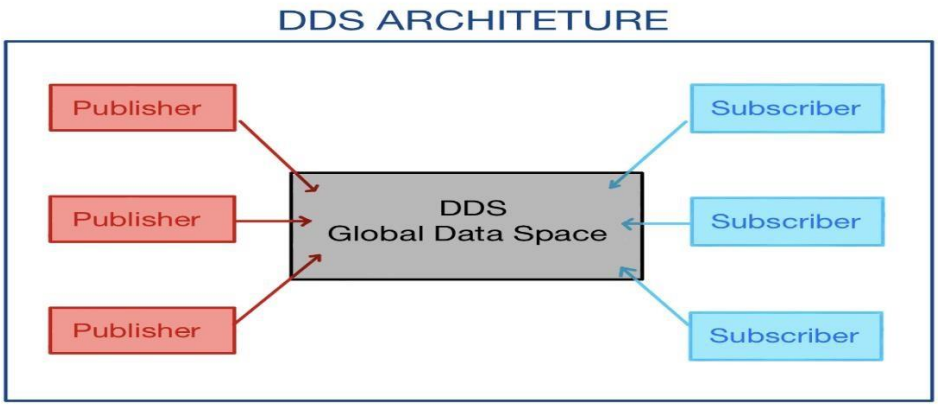


Figure 10 : Operation protocol DDS. [41]

II.3 Comparison of application protocols

"In this section, we will compare the application protocols mentioned previously based on various criteria outlined in

- Protocol standard

- Communication model: Two types of communications:

1. Publish/Subscribe Communication Model: Users can subscribe to specific content to receive real-time updates or notifications.

2. Request/Response Communication: Users can obtain data by sending customized requests with defined messages as per their requirements.

- Transport protocol: Transport layer protocols:

1. TCP: Requires a connection.

2. UDP: Connection is not necessary.

- Security:

1. SSL (Secure Sockets Layer) / TLS (Transport Layer Security): TLS and its predecessor SSL are protocols ensuring the security of exchanges on computer networks, particularly on the Internet.

2. DTLS (Datagram Transport Layer Security): DTLS is designed to protect data privacy by preventing falsification, eavesdropping, and counterfeiting in communications. It is based on TLS, which secures communications networks between computers.

- Main frameworks

- Type of protocol:

There are three types of protocols: messaging protocol, web

transfer protocol, and network protocol."

	XMPP	MQTT	COAP	AMQP	WebSocket	DDS
Standard	The Internet Engineering Task Force (IETF)	Organization for the Advancement of Structured Information Standards (OASIS)	The Internet Engineering Task Force (IETF)	The Internet Engineering Task Force (IETF)	The Internet Engineering Task Force (IETF)	Object Management Group (OMG)
Modèle de Communication	Request /Response Publish/ Subscribe	Publish/Subscribe	Request/Response	Publish/Subscribe	Bidirectional	Publish/Subscribe
Protocol de Transport	TCP	TCP	UDP	TCP	TCP	TCP/UDP
Security	TLS/SSL	TLS/SSL	DTLS	TLS/SSL	TLS/SSL	DTLS/SSL
Framwork	Jabber, XMPPFramework	Emqtt, HiveMQ, Mosquitto, Eclipse Paho	Eclipse Californium, nCoA	RabbitMQ, Storm	jetty websocket, Apache Tomcat	
Protocol type	Messaging	Messaging	Messaging	Web transfer	Network	Messaging

Table II-1:comparison of different application layer protocol [64].

.

The table compares different application layer protocols used in the Internet of Things (IoT) such as XMPP, MQTT, COAP, AMQP, WebSocket, and DDS. It details the standard governing each protocol and the organization managing it, the communication model (like request/response or publish/subscribe), the transport protocol used (TCP or UDP), security mechanisms (such as TLS/SSL), and supporting frameworks or software. These protocols vary in their approach to communication, with some using publish/subscribe or bidirectional communication, and in how they secure data transmission. This comparison helps in selecting the most suitable protocol for specific IoT applications based on communication needs and security requirements.

II.4 Conclusion

In this chapter, we reviewed the various application layer protocols, detailing their characteristics and making a comparison between the most common application protocols. We've also highlighted some key points about how each protocol works.
The next chapter will be dedicated to a simulation study between the two protocols CoAP and MQTT. The objective will be to compare their respective efficiencies and their performances in specific scenarios. This in-depth analysis will allow us to better understand the benefits and limitations of each protocol, helping to guide choices in the Internet of Things (IoT) context.

III. Chapitre 3 : protocoles MQTT et CoAP

III.1 Introduction

IoT protocols encompass various layers including application layer protocols, transport, network, and infrastructure. Within the application layer, several protocols exist such as CoAP, MQTT, AMQP, AXMP, XMPP, and HTTP Rest. Among these, CoAP (Constrained Application Protocol) stands out as one of the most widely utilized protocols. Developed by the IETF specifically for M2M IoT applications, CoAP is tailored for devices with limited resources. It operates on a basis akin to HTTP Rest (Hypertext Transfer Protocol), a clientserver communication protocol originally designed for web applications and adaptable to any reliable connection.

In this chapter, we will delve into the intricacies of CoAP and MQTT protocols, elucidating their respective characteristics and operational mechanisms. Finally, we will elucidate the key distinctions between these two protocols.

III.2 MQTT Protocol:

III.2.1 History

MQTT, originally developed by Dr. Andy Stanford-Clark and Arlen Nipper in 1999, indeed served a crucial purpose in facilitating communication between monitoring devices in the oil and gas industry. The need for a reliable communication method in remote locations where traditional connections were impractical led to the development of MQTT. Its lightweight nature and ability to operate over unreliable networks made it an ideal choice for transmitting

data from thousands of sensors in the field. The standardization of MQTT under the Organization for the Advancement of Structured Information Standards (OASIS) in 2013 further solidified its status as a widely accepted protocol for IoT (Internet of Things) communication. OASIS continues to oversee the development and maintenance of the MQTT standard, ensuring its interoperability and adherence to industry requirements.

III.2.2 Mode of operation

The MQTT protocol operates on a publish/subscribe principle, which differs from the client/server model commonly used on the web. In MQTT, multiple clients connect to a single server, known as the broker, and do not directly communicate with each other. Instead, clients either publish information or subscribe to receive it. Publishers to a channel called a topic send messages. Subscribers, on the other hand, receive messages from these topics. Topics can be organized hierarchically, allowing subscribers to select the specific information they are interested in. This publish/subscribe model enables efficient communication between devices in IoT applications, as it allows for asynchronous messaging and decouples producers of data (publishers) from consumers (subscribers)

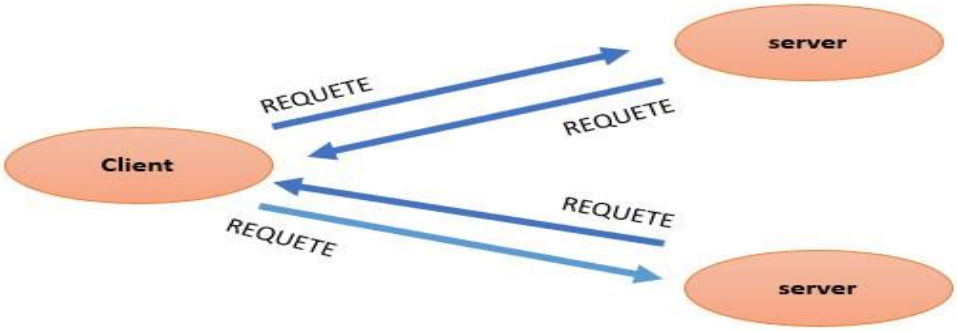


Figure 11 : Client/Server Principle

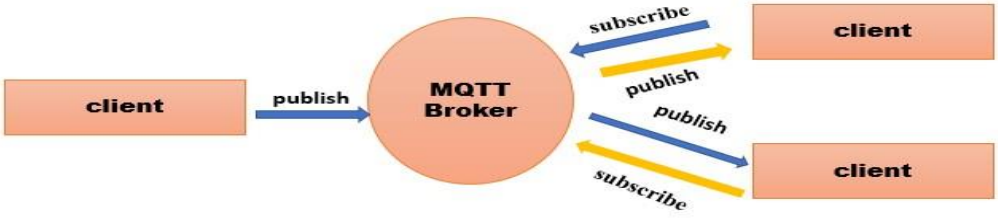


Figure 12 : Principle of operation of the MQTT protocol

III.2.3 MQTT Architect

To understand the MQTT architecture, we first look at the components of the MQTT.

le client: a client in MQTT can be either (subscriber) or(publisher).

b. Broker: the server that manages the transmission of data between clients.

c. Topic: A topic in MQTT or what we call a subject is a point of termination or clients connect. it is a central distribution center for publishing and subscribing messages.

d. the message or information: This is the information we want exchange between devices. it can be either a command or data.

III.2.4 Operation

A. Client Connection and Disconnection:

Connection: Initially, the client registers with the broker using the CONNECT command, facilitating the exchange of connection parameters such as client identifiers. The broker confirms successful registration or indicates an error by returning an error code along with CONNACK message. To maintain the broker's awareness of the client's active status, the client sends periodic PINGREQ commands, eliciting PINGRESP responses from the broker to confirm the ongoing connection.
Disconnection: When the client intends to disconnect, it sends a DISCONNECT command to the broker. This process ensures a structured and secure communication flow without ambiguity.

B. Subscribe and Unsubscribe

●**Subscription:** Clients register with the broker using the SUBSCRIBE command for topics, which serve as access paths to resources. Subscribing clients receive notifications when messages are published on these topics.

●**Unsubscribe:** If a customer wants to cancel a subscription for one or more topics, he uses the UNSUBSCRIBE command and thus he will no longer receive publications that concern these topics. The successful receipt of this command is confirmed by the broker by a UNSUBACK with the same packet ID.

●**Topics:** In MQTT, a "topic" is a UTF-8 string utilized by the broker to filter messages for each connected client. Topics may comprise one or more levels, with each level separated by a forward slash.

Figure 13: topic level separator Compared

to a message queue, MQTT topics are very light.

The customer does not need to create the desired topic before publishing or subscribing to it. The broker accepts each valid subject without any prior initialization.

Examples of topics include:

5ff4a2ce-e485-40f4-826c-b1a5d81be9b6/status: This topic could be used to monitor the status of a specific device or system identified by its unique identifier.de topic allows blank spaces. Subjects are case sensitive.

For example:

_my home / temperature and _My Home / Temperature are two different topics.

In addition, the slash only is a valid subject It is possible to define a tree using the/.

When a customer subscribes to a topic, they can subscribe to the exact topic of a published message or they can use wildcards to subscribe to multiple topics simultaneously. A wildcard can only be used to subscribe to topics, not to post a message. There are two different types of wildcards: single-level and multi-level.

●**Unique level: +** As the name suggests, a wildcard character at a level replaces a subject level. The plus symbol represents a wildcard at a level in a field.

● **Multi Level: #** The multi-level wildcard covers many subject levels. The hash symbol represents the generic multi-level pattern in the subject. For the broker determines the subjects that correspond, the generic character to several levels should be placed as the last character of the subject and preceded slash. **C. The publication:**

The PUBLISH command allows subscribers to post a message that will beset by the broker to potential subscribers. The same order will beset by the broker to subscribers to deliver the message MQTT Message Format an MQTT control packet consists of at most three parts, such as shown in Figure

Fixed header, present all MQTT Control Packets
Venable header, present in some MQTT Control Packets
Payload, present in some MQTT Control Packets

Figure 14 :Structure of an MQTT control packet

.

Fixed header :

Figure 15: Fixed header Format.

• The components of the header:

A. MQTT Control Pocket Types

Name	Value	Direction of flow	Description
Reserved	0	Forbidden	Reserved
CONNECT	1	Client to Server	Connection request
CONNACK	2	Server to Client	Connect acknowledgment
PUBLISH	3	Client to Server or Server to Client	Publish message
PUBACK	4	Client to Server or Server to Client	Publish acknowlegment(QoS1)
PUBREC	5	Client to Server or Server to Client	Publish received(QoS2 delivery part 1)
PUBREL	6	Client to Server or Server to Client	Publish release(QoS 2 delivery part 2)
PUBCOMP	7	Client to Server or Server to Client	Publish complete (QoS 2 delivery part 3)
SUBSCRIBE	8	Client to Server	Subscribe request
SUBACK	9	Server to Client	Subscribe acknowledgment
UNSUBSCRIBE	10	Client to Server	Unsubscribe request
UNSUBACK	11	Server to Client	Unsubscribe acknowledgment
PINGREQ	12	Client to Server	PING request
PINGRESP	13	Server to Client	PING response
DISCONNECT	14	Client to Server or Server to Client	Disconnect notification
AUTH	15	Client to Server or Server to Client	Authentication exchange

Table III-1:Type of MQTT packets [42]

Flags specific to each MQTT Control Packet type: Rema

The inning bits [3-0] of byte 1 in the fixed header contain flags specific to each MQTT control packet type, as noted in the Table III-2 below.

MQTT Control Packet	Fixed Header Flags	Bit 3	Bit 2	Bit 1	Bit 0
CONNECT	Reservd	0	0	0	0
CONNACK	Reservd	0	0	0	0
PUBLISH	Used in MQTTv5.0	DUP	QoS		RETAIN
PUBACK	Reservd	0	0	0	0
PUBREC	Reservd	0	0	0	0
PUBREL	Reservd	0	0	0	0
PUBCOMP	Reservd	0	0	0	0
SUBSCRIBE	Reservd	0	0	0	0
SUBACK	Reservd	0	0	0	0
UNSUBSCRIBE	Reservd	0	0	0	0
UNSUBACK	Reservd	0	0	0	0
PINGREQ	Reservd	0	0	0	0
PINGRESP	Reservd	0	0	0	0
DISCONNECT	Reservd	0	0	0	0
AUTH	Reservd	0	0	0	0

Table III-3:Description of the flag in the fixed header of the MQTT protocol [42]

C. Remaining Length:

The Remaining Length is the number of bytes remaining within the current packet, including data in the variable header and the payload. The Remaining Length does not include the bytes used to encode the Remaining Length.
Here's a breakdown:

- ✓ The Remaining Length field starts at byte 2 of the packet.
- ✓ It's encoded using a variable-length encoding scheme.
- ✓ For values up to 127, it uses a single byte.
- ✓ For larger values, it uses multiple bytes.
- ✓ Each byte contains seven bits for data and one bit as a continuation flag.
- ✓ The maximum number of bytes used for encoding the Remaining Length is four.
- ✓ This encoding scheme allows for efficient representation of packet lengths, especially for larger values, while keeping the overhead minimal. [42]

III.2.5 Variable header:

Certain types of MQTT Control Packets include a variable header component situated between the fixed header and the payload. The specific content of the variable header differs based on the Packet type, but one field, the Packet Identifier, is shared among several packet types.

III.2.6 Payload (charge utile):

a packet may harbor a payload, a component that's both optional and subject to alteration based on the packet type. This segment typically encapsulates the transmitted data, assuming a pivotal role in conveying information. For instance, within the CONNECT packet, the payload

encompasses crucial identifiers such as the client ID, along with supplementary data like "username and password" if provided. Conversely, in the context of a PUBLISH packet, the payload represents the essence of the communication - the message intended for dissemination.

III.2.7 CONNECT:

The following table III-4 is an example of a CONNECT packet [44]:

	Description	7	6	5	4	3	2	1	0
Protocol Name									
byte 1	Length MSB (0)	0	0	0	0	0	0	0	0
byte 2	Length LSB (4)	0	0	0	0	0	1	0	0
byte 3	'M'	0	1	0	0	1	1	0	1
byte 4	'Q'	0	1	0	1	0	0	0	1
byte 5	'T'	0	1	0	1	0	1	0	0
byte 6	'T'	0	1	0	1	0	1	0	0
Protocol Level									
	Description	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 7	Level (4)	0	0	0	0	0	1	0	0
Connect Flags									
byte 8	User Name Flag (1) Password Flag (1) Will Retain (0) Will QoS (01) Will Flag (1) Clean Session (1) *Reserved* (0)	1	1	0	0	1	1	1	0
Keep Alive									
byte 9	Keep Alive MSB (0)	0	0	0	0	0	0	0	0
byte 10	Keep Alive LSB (10)	0	0	0	0	1	0	1	0

Table III-5:format of a CONNECT packet. [42]

Description of the CONNECT packet fields [44]:

Fields	Description
Description	**The connection packet starts with the protocol name, which is MQTT. The length of the protocol name (in bytes) is immediately before the name itself.**
Protocol Level	**Refers to the version of MQTT used, in this case a value of 4 indicates MQTT version 5.0.**
Connect Flags	**Indicate some aspects of the package. For simplicity, this example sets only the Clean Session flag, which tells the client and broker to delete any previous session and start a new one.**
Keep Alive	**The frequency at which the client sends a ping request to the broker to keep the connection active; in this example, it is set to 60 seconds.**

Client ID	**The length of the ID (in bytes) precedes the ID itself. Each client connecting to a broker must have a unique client ID. In the example, the ID is DIGI. When using the Paho MQTT Python libraries, a random alphanumeric ID is generated if you do not specify an ID.**

Table III-6:CONNECT packet fields

III.2.8 CONNACK

The CONNACK header consists of a single bit to indicate whether the broker already has a session for this client, in the event that the client requests to restore an existing session, and a one-byte return code (only codes 0 to 5 are currently defined). [42]

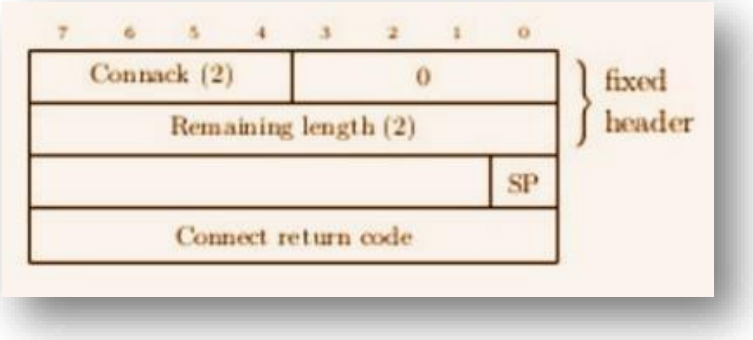


Figure 16:Format Packet CONNACK]43[

III.2.9 PUBLISH

A PUBLISH packet contains the values DUP, QoS, and Retain in its fixed header, followed by the topic (size then non-null string), the packet identifier if the QoS is greater than zero, and finally the message body, which may be empty.

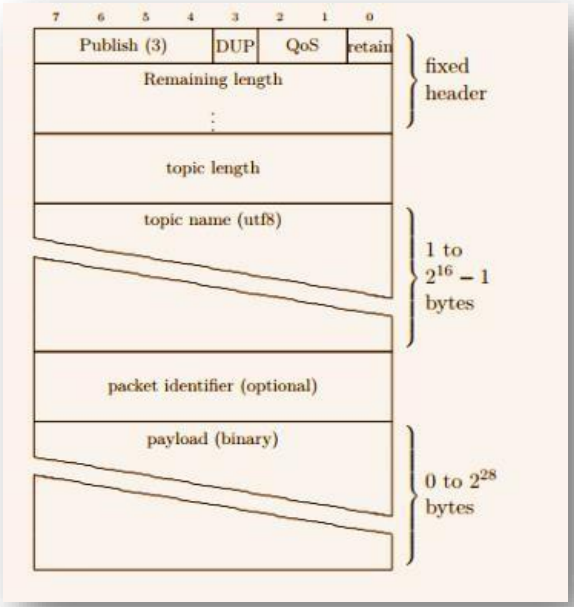


Figure 17:Format of a PUBLISH Packet. [43]

III.2.10 PUBACK, PUBREC, PUBREL, PUBCOMP

All these packages have the same structure, only the values of the order code and the Reserved field can vary.

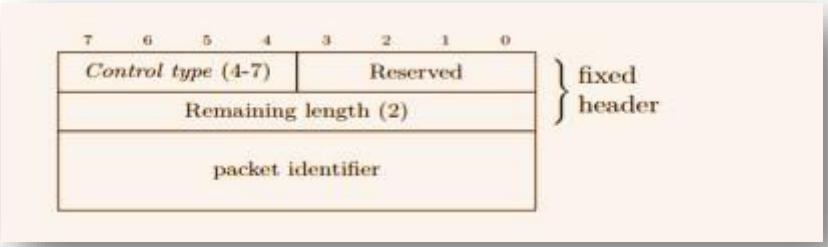


Figure 18:Format of a PUB[ACK/REC/REL/COMP] packet. [43]

III.2.11 SUBSCRIBE

A SUBSCRIBE packet comprises an identifier for the response, followed by a list of subscriptions with at least one element. The structure of a subscription consists of a filter (length and string of characters) and the requested quality of service.

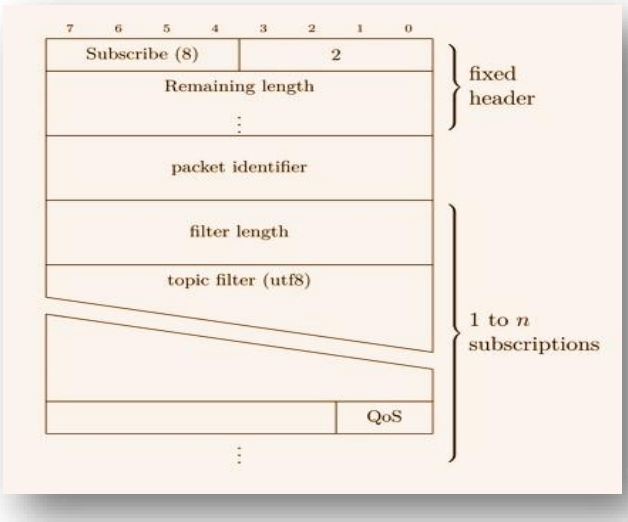


Figure 19:Format of a SUBSCRIBE packet. [43]

III.2.12 SUBACK

In a SUBACK packet, the broker responds with the same packet identifier, then returns the return codes in the same order as the subscriptions in the corresponding SUBSCRIBE packet. These return codes correspond to the quality of service guaranteed by the broker (which may be lower than the initially requested one), or the error code 0x80 in case of an issue. [42]

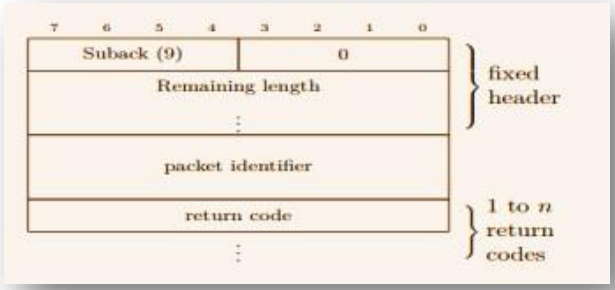


Figure 20:Format of a SUBACK packet. [43]

III.2.13 UNSUBSCRIBE:

UNSUBSCRIBE is very similar to SUBSCRIBE, except that the client does not specify a quality of service.

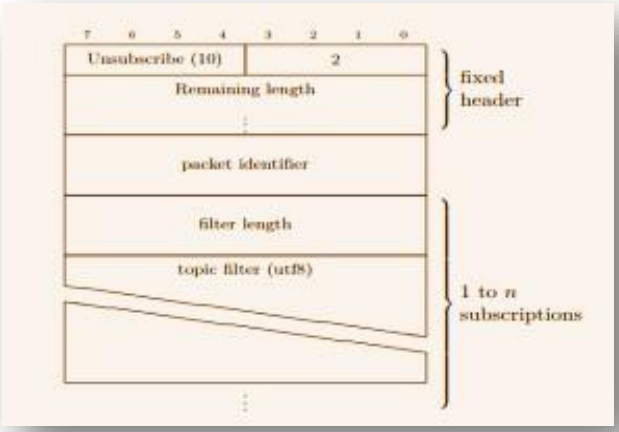


Figure 21:Format of an UNSUBSCRIBE packet[43]

III.2.14 UNSUBACK

UNSUBACK" serves as a confirmation for "UNSUBSCRIBE" and simply contains the packet identifier of the corresponding request.

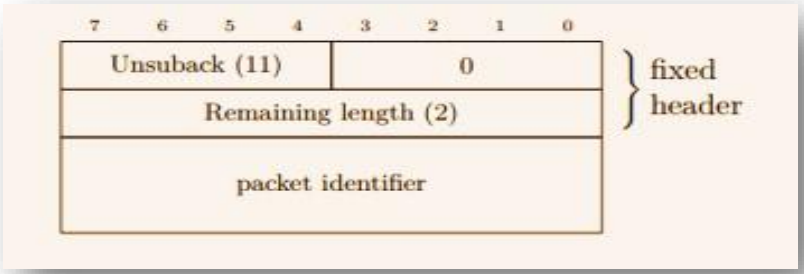


Figure 22:Format of an UNSUBACK packet. [43]

III.2.15 PINGREQ, PINGRESP

These two packets are very simple, and consist only of the fixed header of the protocol, thus taking two bytes.

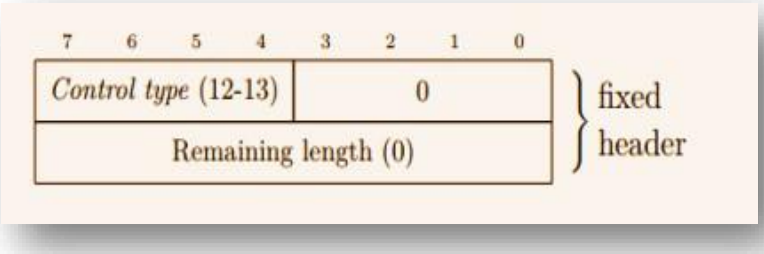


Figure 23:Format of a PING packet [REQ/RESP]. [43]

III.2.16 DISCONNECTION

Used to indicate to the broker the client's intention to disconnect, and only includes the fixed header.

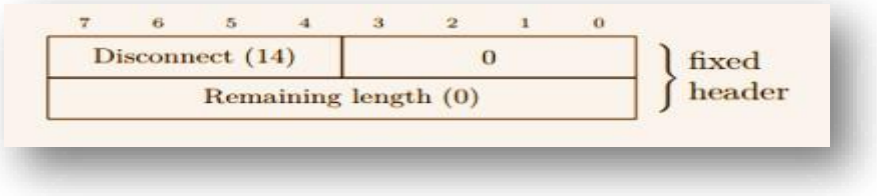


Figure 24:Format of a DISCONNECT packet. [43]

III.2.17 Security

The security of a communication protocol is often essential for many applications, MQTT offers some basic options to help secure applications using this protocol, and we will see that the protocol is flexible enough to integrate other security mechanisms depending on the needs. Furthermore, the possibility of using a TLS/SSL connection allows carry certain guarantees at the connection level, such as authentication server, the infidelity and integrity of the data exchanged. In the end, few security aspects are directly managed by the proto gule so doste to keep a fitotocole the filis sample possible

III.3 The CoAP protocol

III.3.1 Deffinition of CoAP

The Constrained Application Protocol (CoAP), an affordable and straightforward protocol, relies on asynchronous message exchange via UDP. It was defined by the IETF's CoRE Working Group in RFC 7252 and incorporates a lightweight reliability mechanism. Tailored for managing basic resources and devices within networks of constrained nodes, CoAP caters to M2M applications like smart energy and building automation. It bears a striking resemblance to HTTP and is engineered to operate seamlessly on equipment with extremely limited resources.

III.3.2 Features of CoAP:

1. Constrained Web Protocol Meeting M2M Requirements: CoAP is designed to fulfill the needs of Machine-to-Machine (M2M) communication within constrained environments.

2. Asynchronous Message Exchanges: CoAP supports asynchronous communication, allowing devices to exchange messages without waiting for a response before sending the next message.

3. UDP Binding with Optional Reliability: CoAP operates over UDP (User Datagram Protocol) with optional reliability, enabling unicast and multicast requests.

4. Low Header Overhead and Parsing Complexity: CoAP minimizes header overhead and parsing complexity, making it suitable for resource-constrained devices.

5. Simple Proxy and Caching Capabilities: CoAP facilitates simple proxying and caching mechanisms, enhancing scalability and performance.

6. URI and Content-Type Support: CoAP supports Uniform Resource Identifiers (URIs) and Content-Types, enabling seamless integration with existing web technologies.

7. Very Low and Simple Cost to Analyze: CoAP messages are designed to be lightweight and easily analyzed, reducing resource consumption and overhead.

8. Stateless HTTP Mapping: CoAP offers a stateless mapping to HTTP, allowing proxies to provide access to CoAP resources through HTTP interfaces.

9. Security Binding to Datagram Transport Layer Security (DTLS): CoAP provides security features through its binding with Datagram Transport Layer Security (DTLS), ensuring secure communication over unreliable networks.

III.3.3 Mode of operation

CoAP employs a client-server model resembling HTTP, as illustrated in Figure (25), where clients send requests to REST resources to obtain data from a sensor or manage a device and its environment. It is important to emphasize that CoAP facilitates asynchronous exchanges through UDP datagrams.

Figure 25:Principle of operation of the CoAP protocol III.3.3.1

COAP architecture:

The interactive model of CoAP resembles the client/server model of HTTP. Figure illustrates

CoAP's two-layer structure. The bottom layer, known as the Message layer, is tailored to handle UDP and asynchronous switching. The upper layer, the request/response layer, manages communication methods and processes request/response messages [34].

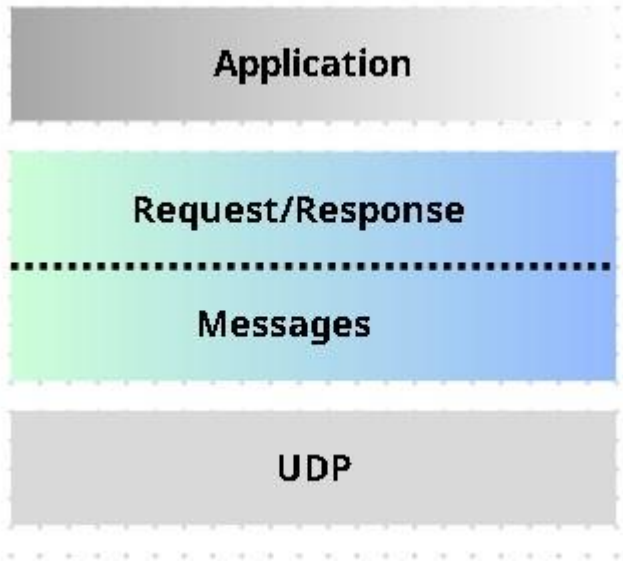


Figure 26:COAP architecture

Message	Description
CON	The confirmable: message necessitates an acknowledgment response from the recipient to ensure reliability functionality.
NON	The non-confirmable: message doesn't necessitate acknowledgment from the recipient.
ACK	Acknowledges the confirmable message.
RST	The reset signal is employed instead of an acknowledgment (ACK) when a Confirmable (CON) or Non-Confirmable (NON) message cannot be processed.

Table III-7: CoAP messaging layer messages

A request is conveyed within either a CON (Confirmable) or NON (Non-Confirmable) message:

Request methods:

The client initiates an action by employing a Method code on a resource identified by a Uniform Resource Identifier (URI) on a server. CoAP specifies four distinct methods:

Method	Description
GET	This method allows you to retrieve information from an identified resource.
POST	This allows you to create a new resource at the requested URL

PUT	This allows you to update the resource located at the requested URL
DELETE	This method deletes the resource in the requested URL

Table III-8:CoAP methods Response

Codes:

The server responds to the client by sending a response code indicating the outcome of the request process. These response codes are divided into three classes, the table 4 explains this:

Code	Beschreibung	Code	Beschreibung
2.01	Created	4.05	Method Not Allowed
2.02	Deleted	4.06	Not Acceptable
2.03	Valid	4.12	Precondition Failed
2.04	Changed	4.13	Request Entity Too Large
2.05	Content	4.15	Unsupported Content-Form
4.00	Bad Request	5.00	Internal Server Error
4.01	Unauthorized	5.01	Not Implemented
4.02	Bad Option	5.02	Bad Gateway
4.03	Forbidden	5.03	Service Unavailable
4.04	Not Found	5.04	Gateway Timeout
		5.05	Proxying Not Supported

Table III-9:CoAP response codes

- A request is transported in either a CON (Confirmable) or NO (Not Confirmable) message:
    1. **Piggy-backed response:**The client initiates a request using either a CON type or NON type message and promptly receives an acknowledgment (ACK) with a confirmable message. In Figure 27, in the event of a successful response, the ACK contains the response message identified by the token. Conversely, in the case of a failure response, the ACK contains a failure response code.

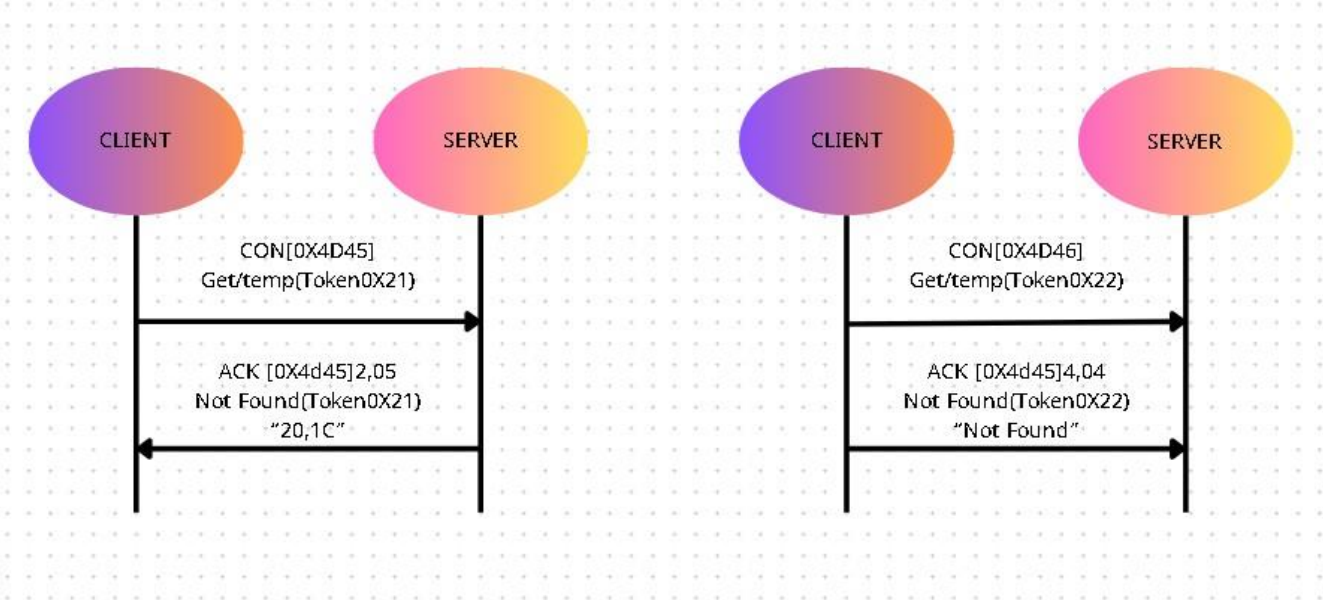


Figure 27:The successful and failure response results of GET method

Separate response: If a server receives a CON-type message but cannot respond immediately, it will send an empty ACK if the client resends the message. Once the server is ready to respond, it will send a new CON to the client, and the client will reply with acknowledgment to confirm the message. The ACK is solely to confirm the CON message, regardless of whether the CON message carries a request or a response (see fig28).

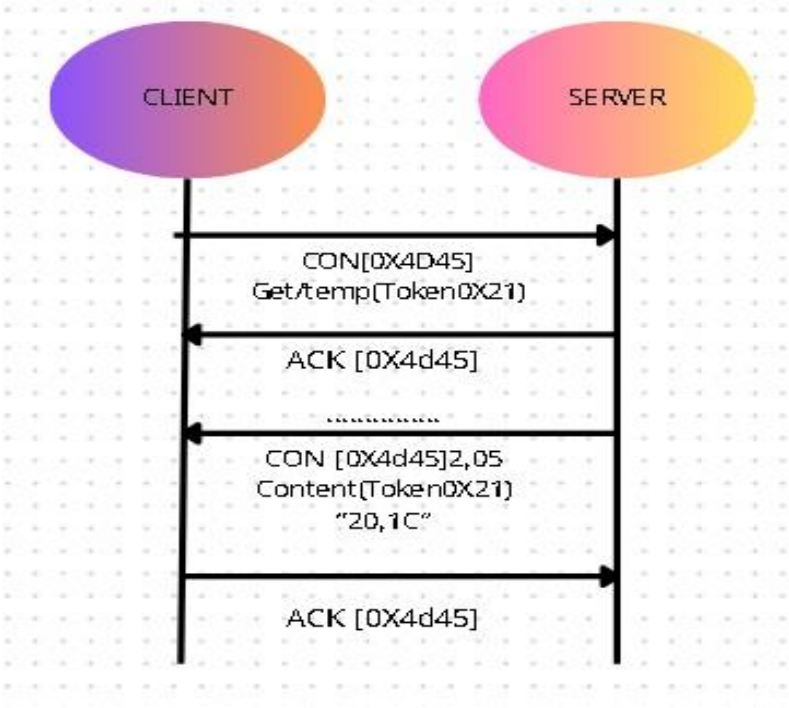


Figure 28:A Get request with a separate response

Non confirmable request and response: Unlike piggybacked responses that carry confirmable messages, in non-confirmable requests, the client sends a NON-type message

indicating that the server does not need to confirm. The server will then resend a NON-type message with the response (refer to).

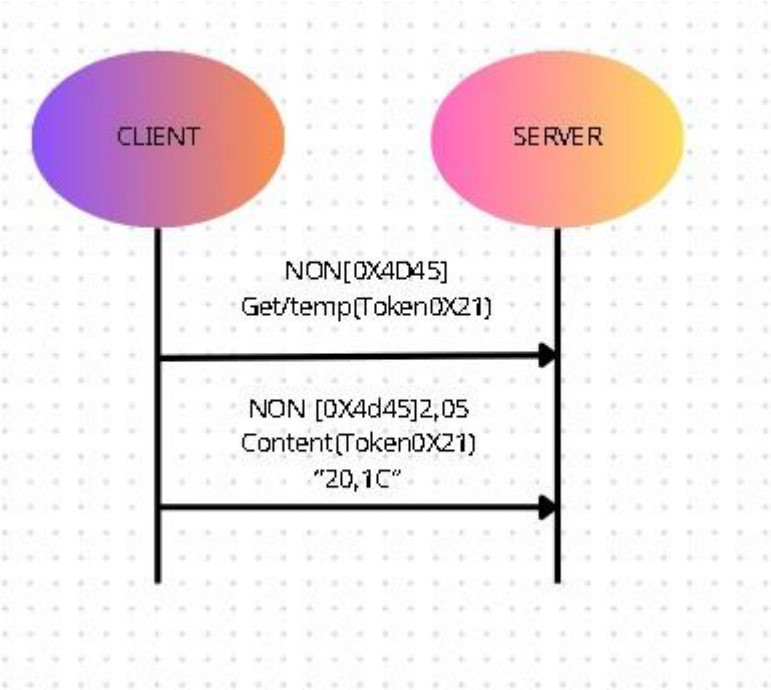


Figure 29:Non confirmable request and response

III.3.4 Message CoAP

The CoAP header was intentionally crafted for straightforward parsing by programs operating on compact devices like sensors (III.20).

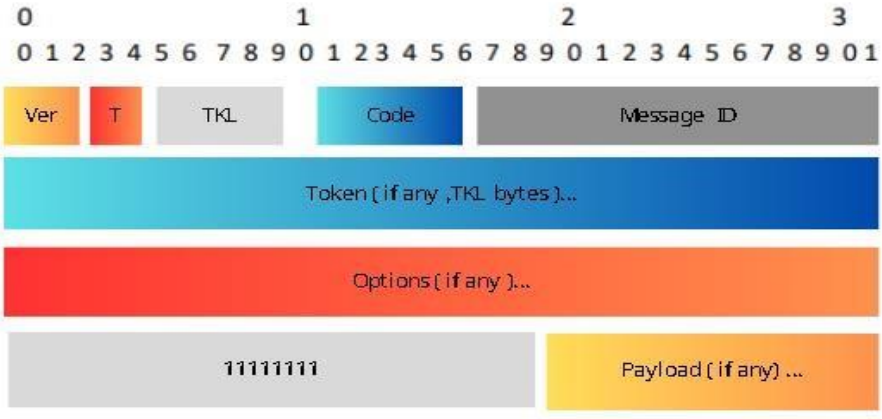


Figure 30:Format du message COAP.

The definition of the message header bits is as follows:

- Version (Ver): a 2-bit integer indicating the CoAP version used.

- Type (T): a 2-bit unsigned integer indicating the message type: Confirmable (0), Nonconfirmable (1), Acknowledgement (2), or Reset (3).

- Token Length (TKL): determines the length of the variable-length token field.

- Code: an 8-bit unsigned integer representing the message code. For Request messages, the range is 1-10, and for Response messages, it's 40-255.

- Message ID: a 16-bit integer used for reliable transmission, duplicate detection, and matching ACK/RST to corresponding CON/NON messages.

- Payload: The payload begins with a one-byte marker, called the Payload Maker, which indicates the start of the payload data. If the Payload Maker has a value of all ones (0xFF16), data is present; otherwise, the payload is empty.

- Options: The options field, if present, contains the option number, option value length, and the value itself. The option number is calculated using the equation: option number = option delta + previous option number. Option delta indicates the difference between the current option number and the previous one. Option length indicates the size of the option value, which can be empty (zero), opaque, unit (option length), or string (UTF-8). The options field has two different classes for handling unrecognized options: critical or elective.

III.4 Comparaisson

III.4.1 Differences between MQTT and CoAP

After examining the various application layer IoT protocols and noting their differences, it is clear that each protocol has its own characteristics to meet the specific needs of the IoT domain. The CoAP and MQTT protocols stand out in particular.

The CoAP protocol is characterized by high speed and efficiency, largely due to the use of UDP, making it a cost-effective solution for IoT communications.

MQTT, on the other hand, features its broker architecture, which simplifies communication management. Additionally, QoS options combined with TCP ensure reliable message delivery, ensuring its reliability in IoT environments.

In conclusion, CoAP and MQTT are preferred for IoT applications as their performance and functionality suit the specific requirements of this domain.

	CoAP	**MQTT**
Transport Layer Protocol	Works over UDP, TCP can be used	Works over TCP, UDP can be used (MQTTSN)
Reliability mechanism	CON, ACK, NON, RST	The 3 levels of quality of service
Communication	Request/Response	Publication/Subscription **model**

Header	4 Bytes	2 Bytes
Messaging mode	Uses both asynchronous and synchronous mode.	Uses asynchronous mode only
Number of message types used	4	16

Table III-10:major differences between MQTT and COAP[63]

III.4.2 Points in common between the two protocols [67] :

- Better suited to constrained environments compared to HTTP.

- Provide asynchronous communication mechanisms, enhancing flexibility.

- Executed over IP, ensuring compatibility with existing infrastructure.

- Offer a range of implementations, catering to diverse needs and preferences.

III.4.3 Conclusion of comparison

After introducing both the CoAP and MQTT protocols and elucidating their respective functionalities, it becomes apparent that while both are valuable for IoT applications, they exhibit fundamental disparities.

The MQTT protocol operates on a publish/subscribe architecture with an intermediary Broker, rendering it well-suited for communication across expansive networks. Its efficiency is particularly evident in scenarios with constrained bandwidth resources.

Conversely, the CoAP protocol distinguishes itself from MQTT by its compatibility with HTTP and its reliance on the UDP protocol. This compatibility enables seamless integration with existing web infrastructures. Moreover, CoAP's utilization of UDP proves advantageous in resource-constrained environments. Notably, UDP facilitates broadcasting and multicasting, allowing for transmission to multiple hosts while conserving bandwidth.

Consequently, CoAP emerges as the preferred choice for local networks wherein rapid interdevice communication is imperative.

III.5 Conclusion

In this chapter, we delineated the distinctive attributes of MQTT and CoAP. We elucidated their operational mechanisms and highlighted the disparities between them. Conclusively, we summarized our findings with a comparative table juxtaposing the two protocols, MQTT and CoAP. Moving forward, the subsequent chapter will entail a comparative analysis conducted through simulation to further explore the nuances between these protocols.

IV. Chapiter 4: Simulation MQTT-SN and COAP

IV.1 Introduction

In this chapter, we will describe the process of carrying out our simulation. We will begin by presenting the development environment and showcasing some interfaces of our simulation. Additionally, we will propose a simulation scenario for both the CoAP and MQTT protocols.

IV.2 Evaluation methods

There are several methods for assessing the performance of a system on a WSN. Analytical modeling, measurements based on real-world experiences, and simulation :

IV.2.1 Analytical Methods

It presents analytical approaches for studying a system's behavior by solving the mathematical equations underlying its mathematical model. Analytical approaches are mostly useful for solving problems that need little computer time. Furthermore, analytical techniques provide for a better knowledge of how a system works since one may evaluate some of its imbalances by solving his model and therefore suggesting changes to address them, such as formal approaches.[57] The analytical technique in scientific writing necessitates evidence, exemplary inspection, and quantitative measurements.

IV.2.2 Real Experience

Validating protocols and applications through real-world tests is complex for several reasons. Firstly, experiments are often difficult to reproduce accurately. Secondly, external factors can disturb the results, and the experimenter has limited control over these variables. Additionally, studying the increase, decrease, and variation in speed and movement patterns is a complex process. One major disadvantage of real-world testing methods is the need to make restrictive assumptions about the actual system to develop workable models. Since our study focuses solely on performance evaluation and does not require real-world application, this method was not chosen. . .[58]

IV.2.3 Simulation

Computer or digital simulation refers to the execution of a computer program on a computer or network. Scientific numerical simulations are based on the implementation of theoretical models. Therefore, they adapt mathematical modeling to digital means and serve to study the functioning and properties of a modeled system, as well as predict its evolution. Graphical interfaces allow users to visualize the results of the calculations.[59]

There are many separate event simulators. Among them we mention the NS-2, NS-3 and OMNeT network simulators, which allow simulation of different types of networks, including queuing networks, as well as OPNET and COOJA is a tool for performance analysis.

COOJA, being the default network emulator for Contiki, was originally compiled with Contiki 3.0. COOJA provides an easy-to-use interface that allows for quick simulation and analysis setup.

While our topic requires performance study, COOJA has proven to be one of the best tools for protocol simulation due to its flexibility, scalability, and rapid prototyping.

IV.3 Tools used in this simulation

In order to achieve a simulation of the COAP protocol we need to know about the used tools.

✓ ContikiOS

Contiki is a lightweight and flexible open-source operating system designed for IoT nodes. It was created by Adam Dunkels and is written in the C language. Contiki enables the connection of small, inexpensive, and low-performing microcontrollers to the Internet, making it suitable for WSN (Wireless Sensor Network) sensors. .[61]

- ✓ **COOJA**

  Is a wireless sensors network simulator depend on Contiki operating system It is a flexible Java-based simulator that supports using C language to develop application software by Java Native Interface. One of the great advantages of this COOJA simulator is that it can simulate the application software simultaneously in highlevel algorithm development and low-level hard driver development. The COOJA simulator has great extensibility. Application developers can alter parts of the simulation environment without changing any COOJA main code. It means that the system can be added to new parts such as interfaces, plugins, and radio mediums or reconfigured existing parts. With these advantages of COOJA, we can implement a variant simulation with different conditions and system settings such as different packet generation rates, different MAC protocols, and different network topology [61].

- ✓ **Python language**

Python is a programming language (like C, C ++, FORTRAN, Java ...), It was developed in 1989. Its main characteristics are as follows: Open source is free to use, source files are available and editable; Equipped with a very extensive base library and a large amount of libraries available for scientific computing, statistics, databases and visualization ... etc. and Dynamic writing is done automatically during execution of the program, which allows great flexibility and speed of programming, but it is motivated by excessive memory

consumption and loss of performance, providing support for "integrating other languages" [59] . It was used in this work to draw curves resulting from the study

✓ **C Language**

C is a compiled language, unlike interpreted languages. A C program is written as a text file called a source file, which cannot be directly executed by the microprocessor. It must be translated into machine language by a compiler. In this work, C was used to modify the files of the COOJA emulator according to the requirements of the study. .[62]

✓ **Java language**

Java is a programming language and computing platform first released by Sun Microsystems in 1995. There are lots of applications and websites that will not work unless you have Java installed, and more are created every day. Java is fast, secure, and reliable. From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere [62]]

IV.3.1 Hardware

- ✓ We used a DELL laptop ✓ RAM: 8.00 GB.
- ✓ System: 64-bit operating system.
- ✓ Windows 10

IV.3.2 The stages of the simulation

Installing the Contiki operating system was the easiest phase. Mosquito installation took significantly longer than anticipated:

```
user@instant-contiki:~$ sudo apt install mosquitto mosquitto-clients
```

IV.3.2.1 Running the Cooja simulator:

The simple way to run Cooja is to run it in its own directory figure (31) cd

contiki/tools/cooja

ant run

```
user@instant-contiki:~$ cd contiki/tools/cooja
user@instant-contiki:~/contiki/tools/cooja$ ant run
```

Figure 31 :Execution of coffee After

running Cooja, the following window appears:

Figure 32:COOJA simulator interface

IV.3.2.2 Creation of a new simulation

In the menu, you must choose: File > New simulation (FigureIV.3). Afterwards, you have to choose a name for the simulation . Then, various windows will appear for the simulation, such as the network window, simulation control window, output mode, and chronology .

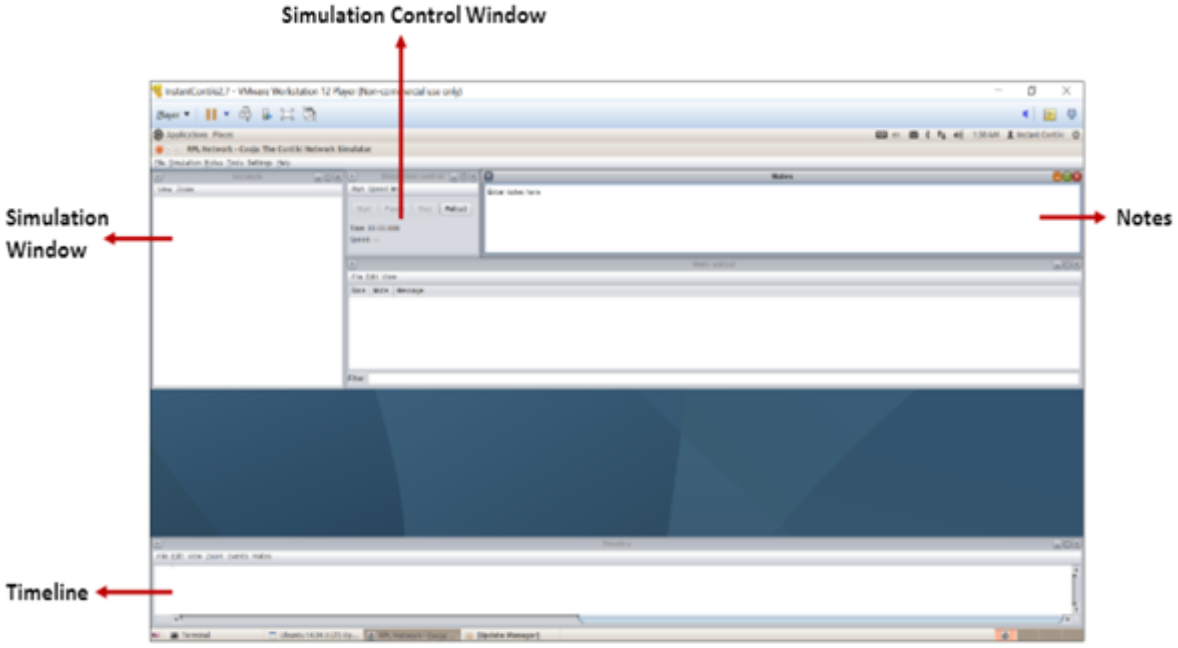


Figure 33:Cooja simulator windows

IV.4 Simulation Environnent

The table (1) summarizes the main parameters of the simulation environment. We are studying the simulation of the protocol. The simulation is adopted by a network of 10 servers. Initially, the nodes are placed in random at 100 x 100 m. The speed of movement of the node varies by a value of 1 m / s. The maximum travel speed and pause time determine how much the model moves. To create a moderate simulation, the pause time is set to 5 seconds. We will simulate different scenarios. Each simulation will take one (1) hour. The default values for some environment parameters are expressed below:

Parameters	Value
Surface	100x100 m
Phase initialization	1 minit
Time simulation	1 hour
Random seed (Random speed)	123,456
Mote startup delay	5s
Speed of nodes	1s
Break time	2s
Number of servers	10

Table IV-1 Simulation Environment

IV.5 Scenario of the CoAP protocol Simulation

Once the new simulation is created and named, we must create the motes we have chosen “Sky Mote” as type motes, we have choose for t the server as quantity (10server), the client (2 clients) and onre router (figure34)

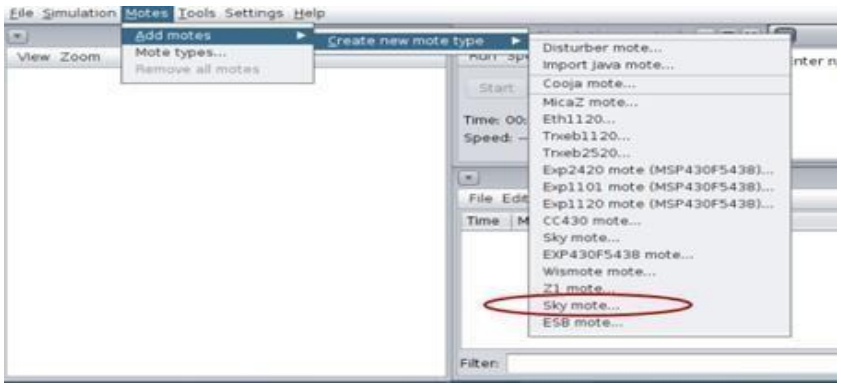


Figure 34: creation of motes

After choosing the type of mote a window appears which allows you to give an ID to the motes and to choose the .C file (by clicking on Browse) which will be compiled as the compilation of a C program and this will allow you to simulate the motes (Figure35)

Figure 35 :Compilation window.

A. The router: We gave it the ID "broder router" and we selected the border-router.c file then we compiled (by clicking on Compile) once the compilation was finished, we chose to create our mote (Create) and another window to give the number of motes that we wish to create and their position appeared we chose 1 and finally we added our router

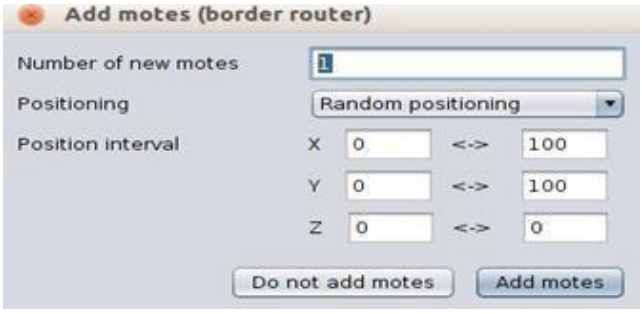


Figure 36 : Adding motes.

- ✓ **Information of sky mote embroider router:**

Figure 37 : sky mote embroider router

B. The server: same thing as the router except that we gave it the ID "Server" and we selected the file er-example-server's

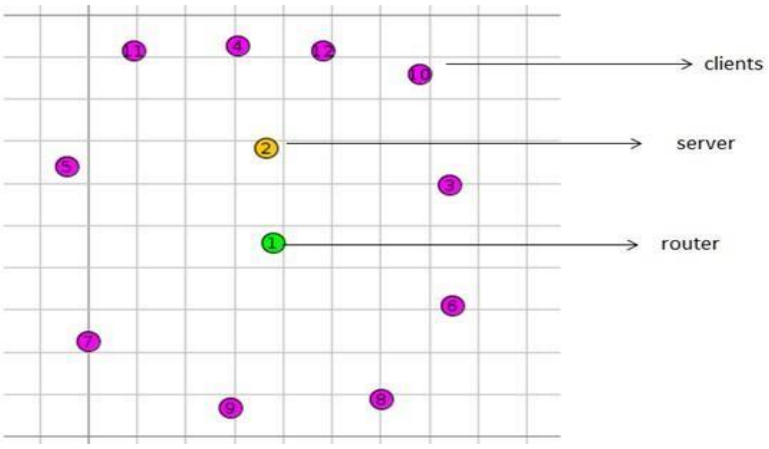
- ➢ **information Sky mote server:**

Figure 38 :Sky mote server

D. Clients: The sky mote for clients was named "client" and the er-example-clients file was chosen. c and in terms of number of motes we chose 10

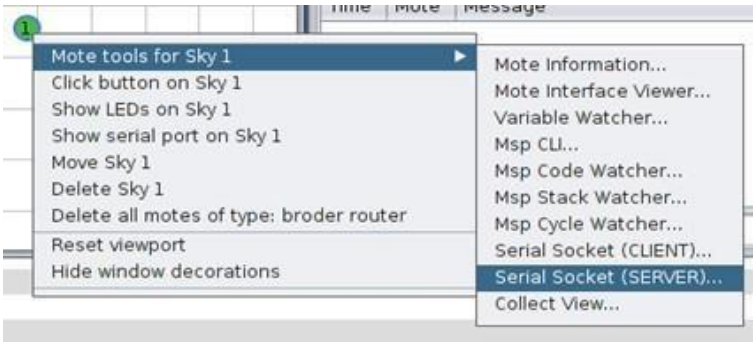
Figure 39 :Sky mote cleint

- This is the result of creating motes:



- **Router:** Information receiver.

Once the motes have been created, the router must be switched on as follows:



✓ After a window with the boot option is displayed

Figure 40 :serial socket server window.

- To start the simulation and allow the client to connect, click (start) in the serial socket server window and the simulation control window

Figure 41 : Starting the simulation.

● Also execute the "make connect-router-cooja" command:

```
user@instant-contiki:~/contiki$ cd examples/ipv6/rpl-border-router
user@instant-contiki:~/contiki/examples/ipv6/rpl-border-router$ make connect-router-cooja
```

IV.6 Simulation of the MQTT protocol

For the MQTT protocol we need 3 motes the router the Publisher and the Subscriber.

A. The router: We gave him the border router ID and selected the file a compiler borderrouter.

Figure 42 :information router

B. Client Publisher: Its ID is "publisher" the compiler file we selected main_core . c.

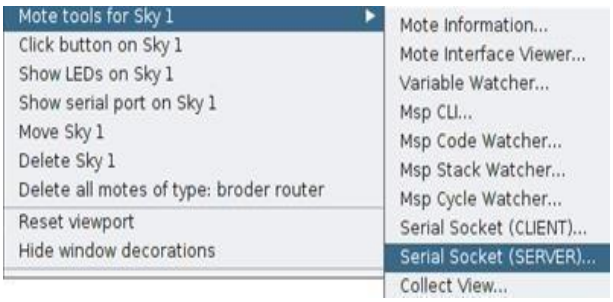
Figure 43 :information Publisher

C. Client Subscriber: Its ID is "subscriber" and for the file a compiler same as the client publisher.

Figure 44 :information Subscriber

- **Publisher:** the server.
- **Border router:** the controller. ● **Subscriber:** . clients.

• Once the motes are created, the router must be turned on:



• Click (start) on the serial server window, then execute the make 'connect-router-cooja' .

- After opening a new terminal and executing the ‘Sodo’ command. /Broker_mqtts config. mqtt.

```
user@instant-contiki:~$ cd contiki
user@instant-contiki:~/contiki$ cd mqtt-sn-contiki/
user@instant-contiki:~/contiki/mqtt-sn-contiki$ cd tools
user@instant-contiki:~/contiki/mqtt-sn-contiki/tools$ cd mosquitto.rsmb/
user@instant-contiki:~/contiki/mqtt-sn-contiki/tools/mosquitto.rsmb$ cd rsmb/
user@instant-contiki:~/contiki/mqtt-sn-contiki/tools/mosquitto.rsmb/rsmb$ cd src
user@instant-contiki:~/contiki/mqtt-sn-contiki/tools/mosquitto.rsmb/rsmb/src$ su
do ./broker_mqtts config.mqtt
```

Figure 45 :command. /Broker_mqtts config. mqtt.

IV.7 Performances evaluation experiments

In order to evaluate the performances of the CoAP and MQTT-SN protocols, we used two parameters in our study (the number of nodes and the interval of data transmission). We examined each parameter in relation to power consumption and throughput. This was done by simulating the MQTT-SN and CoAP protocols and modifying the files provided in the system by adding some code. Each time we changed the number of nodes or the data transmission interval, we collected the results and transformed them into curves for protocol analysis.

IV.7.1 Experiment 1 : Number of servers

In this experiment we will study the throughput, energy consumed according to the number of servers. The first is simulated by 10, then 20, 30, 40 servers, Figures illustrate the simulation according to the number of the server. and data transmission interval 2sec.

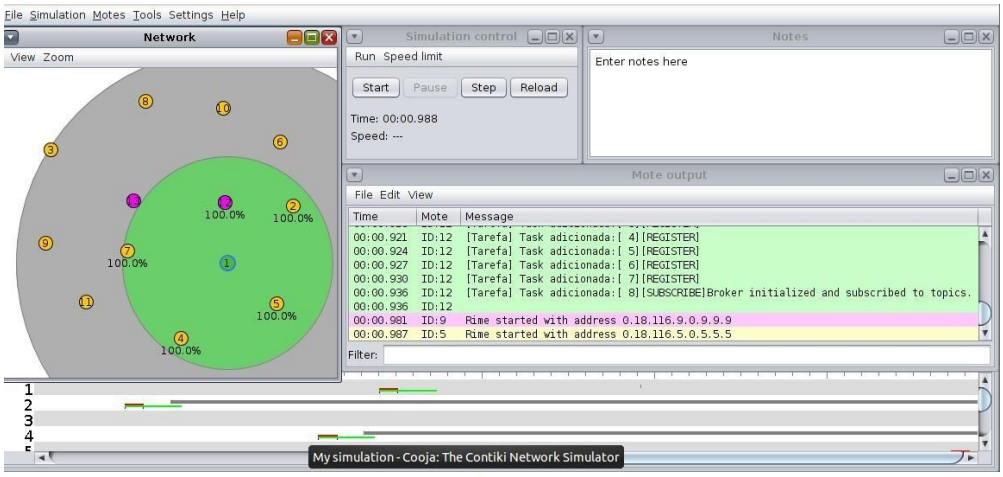


Figure 46 : Simulation For 10 server.

IV.7.2 Experiment 2 : Data Transmission Interval

In this experiment, we will measure what we mentioned earlier according to the field of data transmission interval. We will first take 5 seconds, then 15, 20 seconds, with the number of servers :10 servers

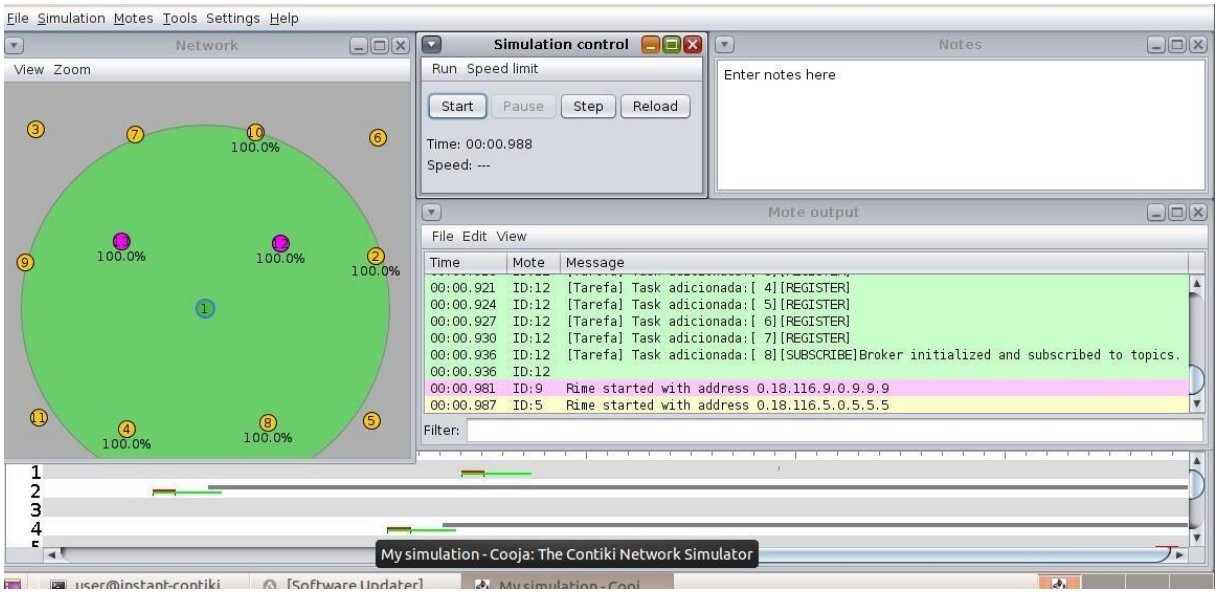


Figure 47Simulation Data Transmission Interval of 2 seconds

IV.8 Conclusion

In this chapter, we provided a brief overview of the steps involved in simulating the CoAP and MQTT protocols. We started by defining the evaluation methods and the various fields in which they are applied. We then presented the programs and the simulation environment used(cooja). We concluded by detailing different simulations. In the next chapter, we will evaluate the performance of both protocols based on these simulations.

V. Chapiter5 :Performances Evaluation

V.1 Introduction

The evaluation of the performance of a system by simulation consists of choosing a model, evaluating it using a simulation technique, and interpreting the collected measurements. Numerous simulation models have been developed to study architectures and protocols under various network scenarios (number of nodes, mobility, etc.). These models have been widely used to evaluate routing protocols.

In this chapter, we will present and discuss the results obtained through simulations in the previous part. Through these results, we will conclude the effectiveness of both the CoAP and MQTT protocols with regard to the Internet of Things.

V.2 Creterias of Evaluation :

To evaluate the performance of the protocol, we study some Measures, including

V.2.1 Network Throughput

The end-to-end network throughput measures the number of packets per second delivered at the destination. It is considered here as an external measure of the effectiveness of a protocol.is calculated as (in Kbps) = (No of successful CoAP request/response pairs * (length of request + length of response in bits)) / total time of simulation.

V.2.2 Energy Consumption

The energy consumption is the sum of used energy of all the nodes in the network, where the used energy of a node is the sum of the energy used for communication, including transmitting (pt), receiving (pr), and idling (pi). assuming each transmission consumes an energy unit, the total energy consumption is equivalent to the total number of packets sent in the network. Power Consumption= (Transmit/19.5 mA + Listen /21.5 mA +CPU power/1.8 mA +LPM/0.0545 mA)/3v/ (32768)

V.2.3 Network Lifetime :

It is considered as the time until the message loss rate is above a given threshold. the more complete definition for the lifetime of the network is "time to network partition" network partition occurs when there is a cut-set in the network. it will be introduced as a new metric, which will use energy variance:

$$\text{network lifetime} = e - (u + \sigma), \text{ where } u = {}_{\sigma}ui/n$$

e is the total initial energy at each node (full battery charge),

ui is the average used energy, **n** is the total number of nodes

in the network, **σ** is expressed

$$\sigma 2 = (ui - u)2\ n$$

All these metrics are calculated using their cumulative average values, that is, at time t, the performance value is the average from 0 to t (seconds)

V.2.4 Packet Generation Rate

It is the number of packets that the sensor node transmits in one time period which is usually one second.

V.2.5 Packet Delivered Successfully

The total number of packets delivered at the destinations versus the total number of packets sent from the source

V.2.6 Latency

The average message latency is defined as the average amount of time between the start of distributing data and its arrival at a node interested in receiving the data. Hence the latency measures time performance for the individual message.

V.2.7 Network Delay

This performance metric is used to measure the average end-to-end delay of data packet transmission. The end-to-end delay implies the average time taken between a packet initially sent by the source, and the time for successfully receiving the message at the destination. Measuring this delay takes into account the queuing and the propagation delay of the packets. It is the sum of 2 * max latency and the processing delay.

V.3 Performance evaluation method

The modifications made to the Contiki files enable us to observe the messages and energy consumption per node through the mote-output window in Cooja. This capability allows us to use these outputs to successfully calculate throughput and energy consumption.

V.3.1 Results Simulation Experiment1: Number of Server

Presents the results extracted from the simulation of the first experiment which is based on the parameter: the number of servers. We calculated the measures of the characteristics (throughput, energy consumption) in each time (10,20,30,40 server), By keeping the same simulation surface, and Transmission interval which is 2sec.

We see in Figure 48 an acceptable throughput value when taking 10 servers, and then at

The number of servers increases gradually, and the transfer rate increases slightly

✓ Figure 48 shows the throughput values in relation to the number of servers

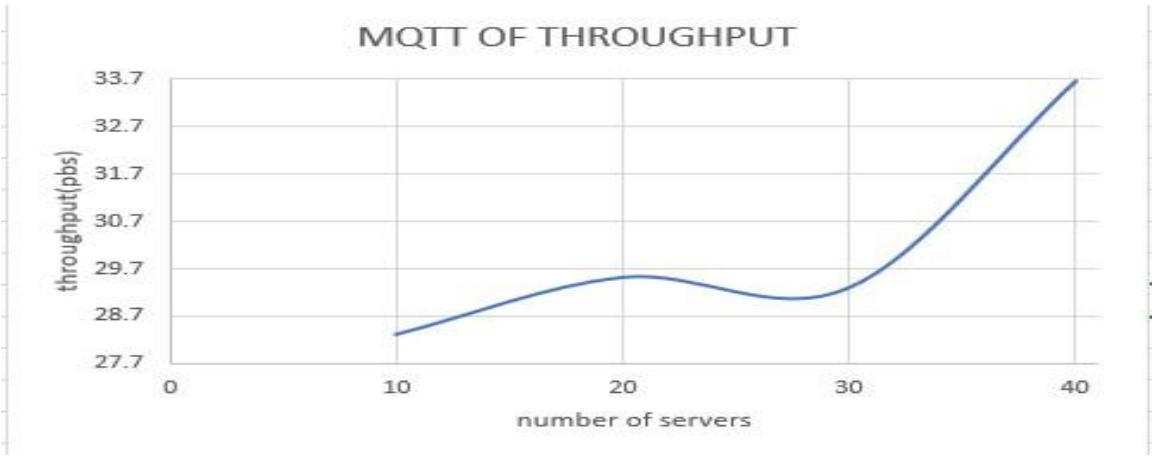


Figure 48 :Curves of throughput(pbs)against the Number of MQTT server

We notice in the figure 48 an acceptable throughput value when taking 10 servers, then when the number of servers increases gradually, the throughput increases successively from 20 bits per second to 40 bits per second.

✓ Figure 49 shows the throughput values in relation to the number of servers

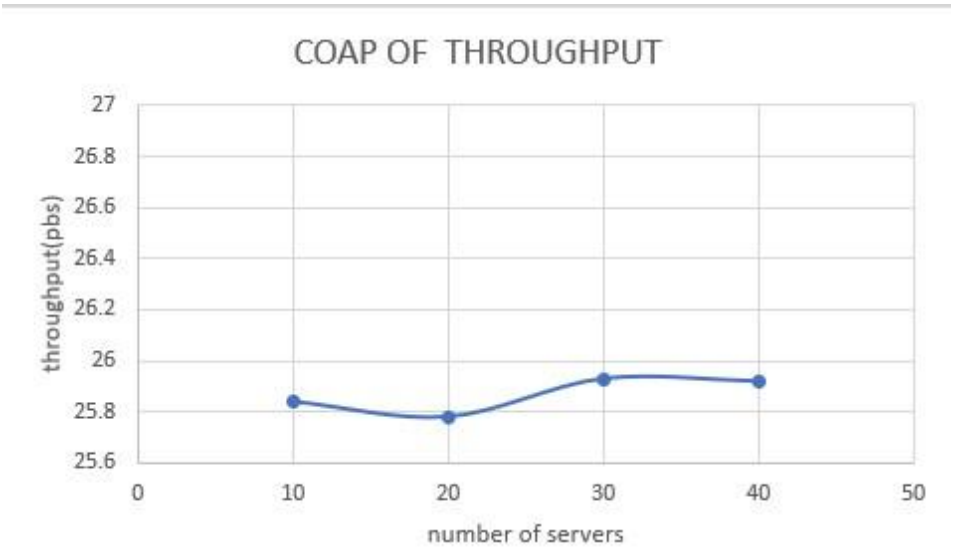


Figure 49Curves of throughput(pbs)against the number of COAP server

We notice in Figure 49 an acceptable throughput value when using 10 servers. As the number of servers gradually increases, the throughput value also increases significantly.

✓ Figure 50 Shows the energy consumption values in relation to the number of servers:

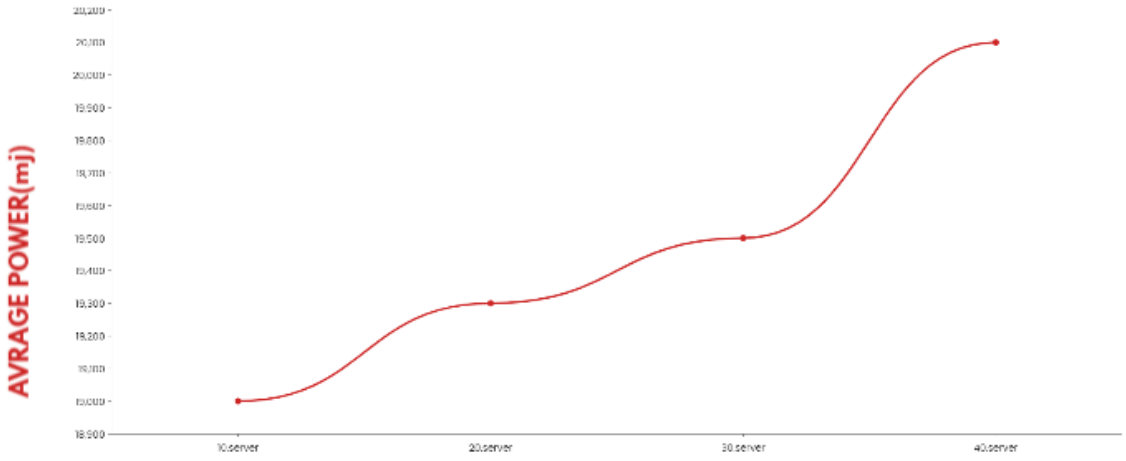


Figure 50 :Curves of energy consumption(mj)against the Number of COAP server

In figure 50 we notice the value of energy consumption when consuming 10servers a few, then it increases with the number of servers gradually increasing

✓ Figure 51 Shows the energy consumption values in relation to the number of servers:

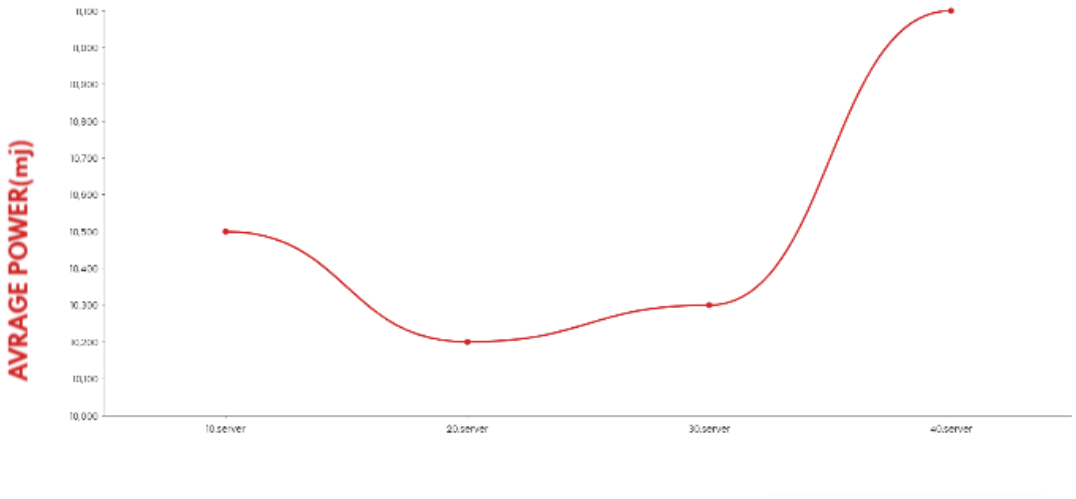


Figure 51 :Curves of energy consumption(mj)against the Number of MQTT- SN server

There is a slight decrease in average energy between the first and second configuration (from 2C10smqtt to 4C20smqtt).
After that, the average energy remains relatively constant between the second and third configuration (from 4C20smqtt to 6C30smqtt).
A significant increase in average energy is observed in the last configuration (12C40smqtt).

- ✓ **Analyze**

When studying throughput, as well as energy consumption, we note with a gradual increase in the number of servers:

- ✓ when throughput depends on the number of servers, increasing the number of servers leads to an increase in throughput.

  when energy consumption depends on the number of servers, increasing the number of servers leads to an increase in energy consumption.

V.3.2 Results Simulation Experiment 2: Data Transmission Interval

Show results extracted from the second simulation Parameter-driven experiment: data transfer interval. We did the math Characteristic metrics (throughput, and power consumption) in each Time (5 seconds, 15 seconds, 20 seconds), by keeping the same simulation surface, and the number of servers is 10Servers.

- ✓ figure 52 shows the throughput values in relation to the data transmission interval

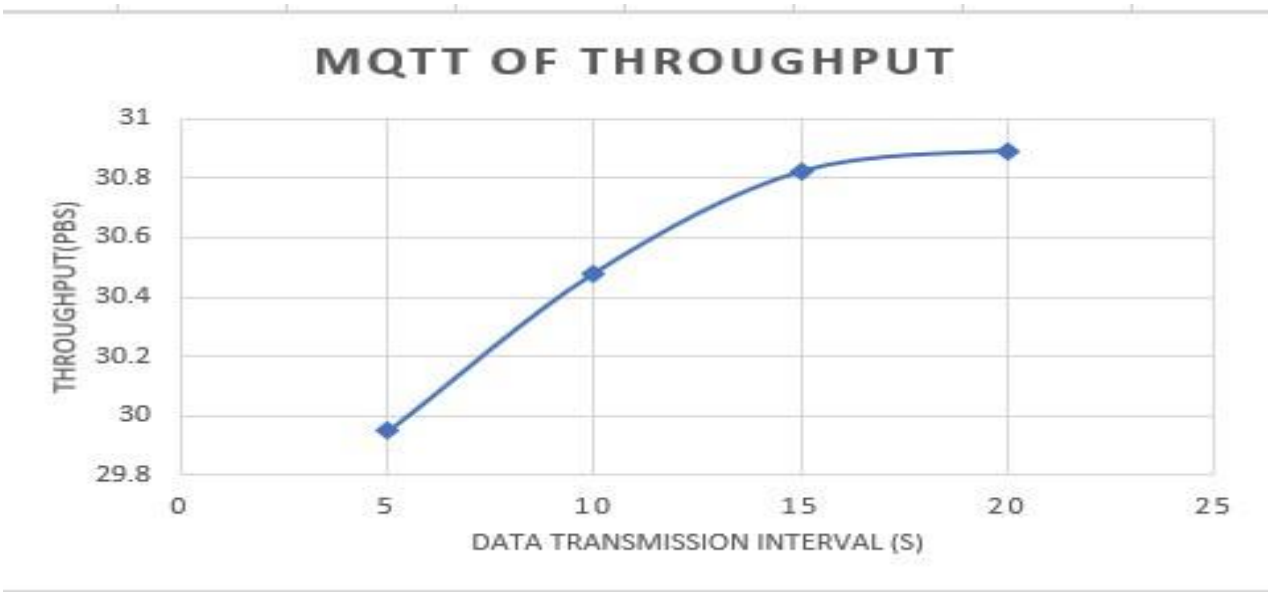


Figure 52 : Curves of throughput(pbs) against the Data transmission interval (s)MQTTSN

We notice in Figure 52 that when the access time exceeds 5 seconds, the throughput period increases significantly.

- ✓ figure 53 shows the throughput values in relation to the data transmission interval

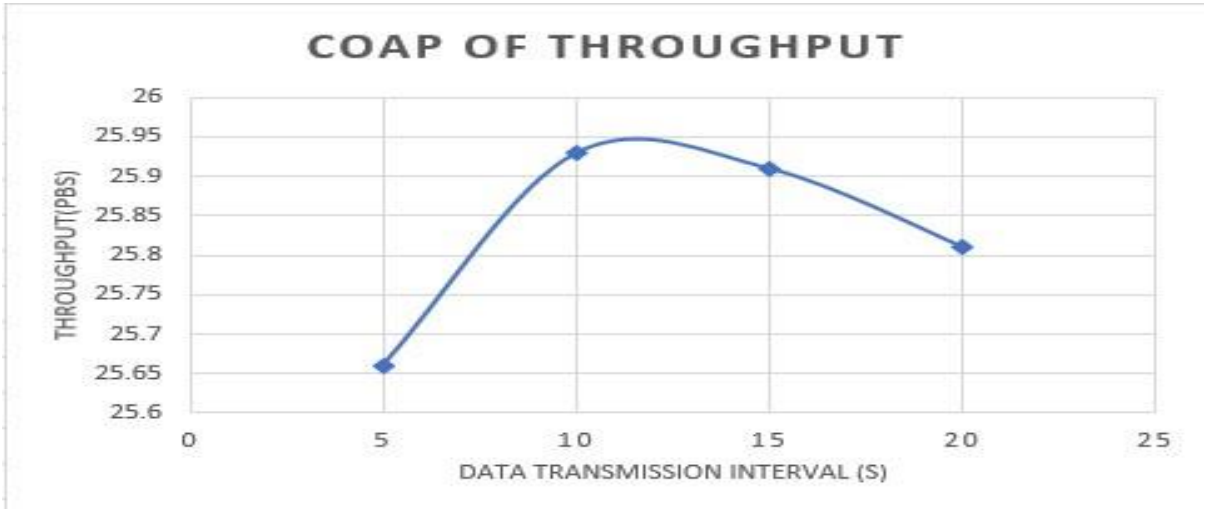


Figure 53 :Curves of throughput(pbs) against the Data transmission interval (s)

We see in Figure 53 that when the access time exceeds 5 seconds, the throughput period decreases significantly

- ✓ The figure 54 shows the energy consumption values in relation to the data transmission interval

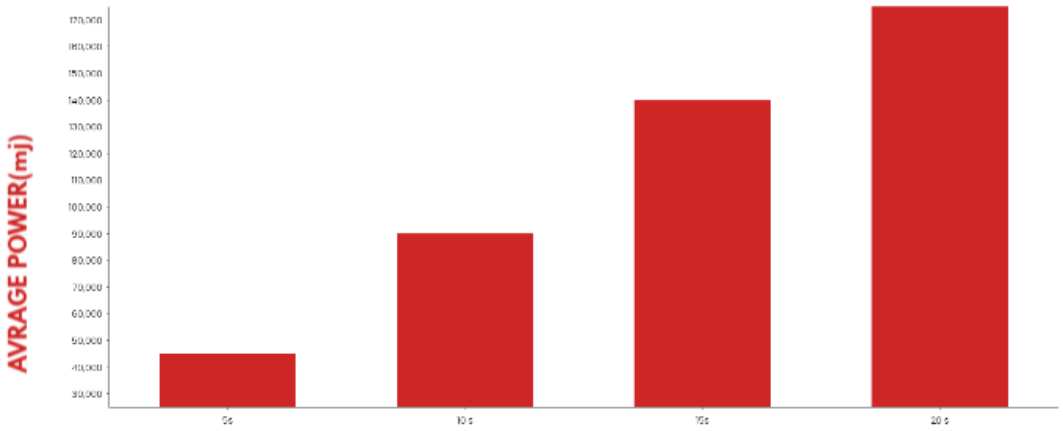


Figure 54 :Chart columns of energy consumption(mj) against the Data transmission interval *(s)*

In Figure 54 we see the power consumption value when the access period exceeds 15 seconds

Increased significantly

- ✓ The figure 55shows the energy consumption values in relation to the data transmission interval

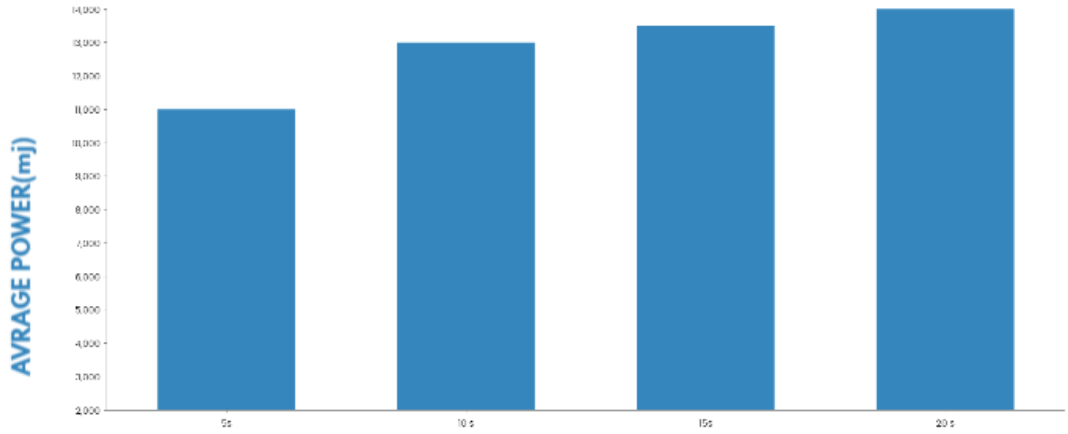


Figure 55 :Chart columns of energy consumption(mj) against the Data transmission interval (s)

In Figure 55we see the power consumption value when the time period is 5 seconds it increased dramatically.

- ✓ **Analyze**

When studying throughput, as well as energy consumption, we note with a gradual increase in data transmission interval:

- ✓ Increasing the data transfer interval means increasing power consumption
- ✓ Increasing the data transfer interval means increasing throughput

V.4 Evaluation results

After simulations performed with COOJA and analyzing the results shown in the form of a graphical curve, the results were arrived at for performance evaluation

- ➢ Throughput decreases as the number of servers increases: This is mainly due to network congestion and resource contention. As more servers are added, the network becomes more congested, leading to more collisions and retransmissions of data, which reduces throughput.

- ➢ Power consumption increases as the number of servers increases: As the number of servers increases, each device has to handle more traffic on the network and spend more time processing and sending data, which increases power consumption.

The design of MQTT-SN focuses on reducing overhead and improving energy efficiency, making it slightly more efficient in scenarios with a large number of servers.

To compare CoAP (Constrained Application Protocol) and MQTT-SN (Message Queuing Telemetry Transport for Sensor Networks) in terms of energy consumption and throughput, especially as the number of servers increases, several factors come.

Comparison with Increasing Number of Servers

Energy Consumption:

- **CoAP:** Energy consumption may rise with the number of servers due to the increased number of direct interactions and potential retransmissions for confirmable messages. Each server needs to handle its communication independently.
- **MQTT-SN:** Energy consumption remains relatively stable due to the broker's role in managing communications. Devices can remain in low-power modes and rely on the broker for efficient message distribution.

Throughput:

- **CoAP:** Throughput may decline as the number of servers increases, especially if many servers are attempting to communicate simultaneously. The synchronous nature and direct communication can be affected.

- **MQTT-SN:** Throughput tends to scale better with the number of servers, as the broker can efficiently manage and route messages. The publish/subscribe model inherently supports higher volumes of message traffic without direct device-to-device communication.

V.5 Conclusion

In this chapter, we presented an illustrative study to evaluate the performances of the two protocols (CoAP and MQTT sn) through measurements extracted from various simulations, and through the results we concluded that, both the energy consumption and the throughputs increases but in different close measurement for the two protocols when increasing the number of servers or the data transmission interval.

Conclusion

The Internet of Things (IoT) represents one of the most significant innovations today, making it crucial for its products to be reliable and effective while adhering to all necessary characteristics. In this research, we concentrated on the protocols due to their variety and the challenge of selecting the best one for users. Our focus was specifically on evaluating CoAP performance through simulations using the internal COOJA network simulator under very complex conditions to assess this protocol's effectiveness.

We employed several tools in this study, as the COOJA simulation. COOJA, based on ContikiOS operating system, which facilitated the connection between clients and servers. This setup allowed us to study and evaluate the protocols based on criteria such as energy cosumptionand throughput, and productivity, to determine its suitability for IoT characteristics. Our work was conducted in five stages:

- Phase 1: Conducted a general study on the field of the Internet of Things, including its advantages, uses, characteristics, and protocols.
- Phase 2: Examined the concept of protocols and the functionality of each protocol.
- Phase 3: Analyzed how the MQTT-SN and CoAP protocols operate within the COOJA simulator.
- Phase 4: Simulated the MQTT-SN and CoAP protocols.
- Phase 5: Efficacy and outcome studies have been conducted, evaluating the MQTT-SN and CoAP protocols.

The results show that the total power consumption is close and the throughput is close too , with MQTT_SN being slightly more efficient than CoAP when we increase the number of servers.

- **CoAP:** Best suited for scenarios with a lower number of servers and simpler, less frequent communications. It is energy efficient for direct interactions but may face challenges in scalability and throughput as the number of servers increases.
- **MQTT-SN:** More suitable for larger-scale deployments with frequent and numerous message exchanges. The broker-based architecture and support for

sleeping clients make it both energy efficient and capable of maintaining high throughput as the number of servers grows.

Choosing between CoAP and MQTT-SN depends on the specific requirements of your IoT application, including the expected number of servers, communication patterns, and energy constraints. For applications needing high scalability, MQTT-SN is often the better choice. For simpler, direct interactions with fewer devices, CoAP can be more appropriate.

Through the end of our study, we have to think about some suggestions that can be added in the future, including:

- Evaluating the performance of CoAP and mqtt-sn with features we did not use as delay and packets loss.
- Proposition of an Improvement in the CoAP or MQTT sn protocols, so that it becomes the primary protocol in the internet application layer in IoT.

Bibliography

[01] P-J. Benghozi, S. Bureau, F. Massit-Folléa, C. Waroquiers, and S. Davidson. L'internet des objets : quels enjeux pour l'Europe. Éd. de la Maison des sciences de l'homme, 2009.

[02] Tafazolli, R. Technologies for the wireless future. Chichester: Wiley, 2006.

[03] Taleb Omar, Mankouri Abdelkrim. « Programmation de la sécurité Internet des Objet, Etude de cas module WIFI Electric imp », Mémoire de master, Université de Tlemcen, Algérie, 2016.

[04] Yacine Challal. Sécurité de l'Internet des Objets : vers une approche cognitive et systémique. [en ligne]Réseaux et télécommunications [cs.NI]. Université de Technologie de Compiègne, 2012. Disponible sur (Consulted on March 9, 2024)

[05] Atoumi .M Y, Bensadi. S, « Approche évolutionnaire pour la composition de services sensible à la QoS dans l'Internet des Objets à large échelle », Mémoire de master, Université de Bejaia, Algérie, 2018

[06] https://www.digora.com/fr/blog/definition-iot-et-strategie-iot(Consulted on March 09, 2024)

[07] Yick, J., Mukherjee, B. and Ghosal, D., Wireless sensor network survey. Computer Networks, 52(12), pp.2292-2330, 2008.

[08] Connectwave. 2022. Comment se compose un système IoT ? [En ligne] Disponible à : (Consulted on February 4, 2024).

[09]

[10] Stankovic, J. Wireless Sensor Networks. Computer, 41(10), pp.92-95, 2008.

[11] S. Rabeb, « Modèle collaboratif pour l'Internet of Things (IoT), » Mémoire de maitrise université de Québec à Chicoutimi, 2016.

[12] Saleh, I. Internet des Objets (IdO) : Concepts, Enjeux, Défis et Perspectives. Internet des objets, 2018.

[13] MELITI Nedjema. Architecture Basée Agents pour le diagnostic d'un système d'IoT (Internet ofThings). [en ligne]. Mémoire de Master académique, Architectures Distribuées. Oum-El-Bouaghi : UniversitéLarbi Ben M'hidi Oum-El-Bouaghi, 2017. Disponible sur : (Consulted on March 9, 2024)

[14] https://www.synox.io/4-choses-a-savoir-sur-linternet-des-objets/ (Consulted on March 9, 2024)

[15]

[16]

[17] PradyumnaGokhale, OmkarBhat, SagarBhat," Introduction To IoT",International Advanced Research Journal In Science, Engineering And Technology,January 2018

[18]

[19]

[20]

[21]

[22] arXiv:2312.06689v2 [cs.CR] 12 Jan 2024

[23]

[24]

[25]

[26]

[27]

[28]

[29] https://fr.wikipedia.org/wiki/VMware_Workstation_Pro (Consulted on April 12, 2024)

[30]

[31]

[32]

[33] https://www.webchoiceonline.com.au/fundamental-characteristics-that-makestheinternet-of-things-what-it-is/ (Consulted on April 15, 2024)

[34]

[35] http://www.wikiwai.com/2020/02/21/internet-des-objets-architectures-protocolesetapplications (Consulted on April 15, 2024)

[36] https://www.educba.com/introduction-to-iot/ (Consulted on April 15, 2024)

[37] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of things : A survey on enabling technologies, protocols, and applications.[en ligne].IEEE Communications Surveys & Tutorials, 2015. Disponible surhttps://ieeexplore.ieee.org/document/7123563

[38] S.Feng, J.Cerles, H.Dalmas, T.Do-Khac, and B. Paulin. Sécurité des objets Connectés. [en ligne]Institut national des hautes études de la sécurité et de la justice, 2014. Disponiblehttps://www.cigref.fr/

[39]

[40] https://www.emnify.com/iot-glossary/guide-iot-protocols (Consulted on April 15, 2024)

[41] Publicis Sapient France – Medium.

[42] https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html (Consulted on April 15, 2024)

[43] https://www.slideshare.net/ymelka/model-based-testing-des-applications-du-protocolemqtt (Consulted on May 15, 2024).

[44] https://www.linkedin.com/pulse/mqtt-50-control-packet-explained-01-connect-connackemqtech (Consulted on April 15, 2024)

[45]
[46]
[47]
[48]
[49]
[50]
[51]
[52]
[53] https://fr.wikipedia.org/wiki/Simulation_informatique (Consulted on May 10, 2024)
[54] http://www.contiki-os.org (Consulted on May 10, 2024)
[55] Edosoft Factory, CONTIKI AND TINYOS, August 13, 2012
[56] Majda Moussa. "Vérification Et Configuration Automatiques De Pare-Feu Par Model Checking Et Synthèse De Contrôleur", Université De Montréal, Mémoire De Maitrisées Sciences Appliquées, P 6, 2014.
[57] WassilaKorichi, "Partage De Données En Environnements Mobiles Ad Hoc", Magistère En Informatique, UniversiteKasdi Merbah Ouargla, P92.
[58] "Introduction A La Programmation En Langage Python", Université Paris-Sud, Méthodologie Licence Mpi S2 - Année 2015-2016
[59] www.eclipse.org (Consulted on May 20, 2024).
[60] L. B. Saad, C. Chauvet, And B. Tourneau, "Simulation of The Rpl Routing Protocol for Ipv6 Sensor Networks: Two Cases Studies," In Proc. Of The 2011 International Conference on Sens or Technologies and Applications (Sensorcomm'11), Nice, France. Iaria, September 2011.
[61] www.java.org (Consulted on May 20, 2024).
[62] https://www.rfc-editor.org/info/rfc9431 (Consulted on May 20, 2024).
[63] https://www.rfc-editor.org/ (Consulted on May 25, 2024).