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

Memoire

With a view to obtaining the diploma of

MASTER Informatique

(Specialty: Network administration and security)

Presented by:
KHALED KHODJA ELYES

BENCASI MONSAF

Comparative Analysis and dataset generation for evaluating IoT Protocols: MQTT-SN , CoAP

Encadreur : KHELILI Khalida Farida
AFRAH Djeddar

Annie universitaire :2025/2024

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

وَأَن لَّيْسَ لِلإِنسَانِ إِلَّا مَا سَعَى اية 39

وَأَنَّ سَعْيَهُ سَوْفَ يُرَىاي اية 40

صَدَقَ ٱللهُ العظيمُ

الاهداء

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.

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 (مباركة)الحبيبة حفظها الله.

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

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 أختي "نادية " اخواني " عبد الحق " عمر" عبد الرحيم" ياسين "يوسف " يونس"حفضهم الله

كل من كان عونا وسنداً في هذا الطريق .. للأصدقاء الأوفياء ورفقاء السنين وأصحاب الشدائد والأزمات إلى من أفاضني بمشاعره ونصائحه المخلصة "اسلام قادري" صلاح بيات" هشام بن لخضر "إسكندر مفتاح"

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 مبارك أينما كنت .

خريجكم الياس

الإِهْدَاء

إلى من كانت لي وطنًا، وسندًا، وملجأً عند الضيق .. من كانت دعواتها زادي في كل طريق، وابتسامتها تبعث في نفسي الطمأنينة والأمان...

أهديك هذه المذكرة راجياً من الله عزّ وجلّ أن يجعلها صدقة جارية في ميزان حسناتك، وأن يتقبل مني كل خطوة خطوتها في طريقي العلمي بنية البر بكِ، والدعاء لكِ، والوفاء لذكراك.

إلى روحك الطاهرة يا أمي، أهديك ثمرة هذا الجهد، ونتيجة أعوامٍ من التعب والسهر، التي ما كنت لأصل إليها لولا غرسك الطيب، وتربيتك الصالحة، وعطائك الذي لا يُضاهى.

إلى أبي، عافاه الله و هداه ، إلى إخوتي الأعزاء حمزة، سفيان، أسامة، وليد، الذين كانوا دومًا سندي في دربي، وتعلمت منكم الصبر والقوة
إلى أخواتي الغاليتين نادية ورميصة، اللتين لم تبخلا عليّ بالدعاء والكلمة الطيبة.

إلى أستاذتي الفاضلة خليلي فريدة، التي كان لتوجيهها ودعمها العلمي والمعنوي أثر بالغ في مسيرتي.

إلى كل من وقف إلى جانبي، وقدم لي يد العون والمساندة، مهما كانت صغيرة، في سبيل إنجاز هذا العمل،، أُهدي هذا التتويج بكل امتنان.

﴿رَبِّ ارْحَمْهُمَا كَمَا رَبَّيَانِي صَغِيرًا﴾

خريجكم منصف

Abstract

The rapid growth of the Internet of Things (IoT) has intensified the need for efficient communication protocols in constrained environments. This study evaluates two key IoT protocols, MQTT and CoAP, through systematic simulations in COOJA/Contiki OS, analyzing their energy efficiency, latency, throughput, and reliability under varied network conditions.

We generate a machine learning-ready dataset capturing protocol performance across different payload sizes, transmission intervals, and network scales. This dataset can be used to enables predictive analysis of protocol behavior and supports data-driven IoT system design.

Results show MQTT achieves better energy efficiency and scalability for large networks, while CoAP excels in low-latency scenarios. Beyond performance comparison, this work provides: (1) a methodological framework for protocol evaluation, and (2) a structured dataset for future IoT research, bridging protocol analysis with machine learning applications.

The combined results and dataset offer practical insights for protocol selection in smart infrastructure and industrial IoT deployments, where constrained resources demand optimized communication solutions.

Keywords: Internet of Things, IoT protocol, machine learning, CoAP, MQTT-SN, Dataset , Performances evaluation, simulation, COOJA.

الملخص

أدى النمو السريع لإنترنت الأشياء (**IoT**) إلى تكثيف الحاجة إلى بروتوكولات اتصال فعّالة في البيئات المحدودة. تُقيّم هذه الدراسة بروتوكولين رئيسيين لإنترنت الأشياء، هما **MQTT** و**CoAP**، من خلال عمليات محاكاة منهجية في نظامي التشغيل **COOJA/Contiki**، مع تحليل كفاءتهما في استخدام الطاقة، وزمن الوصول، والإنتاجية، والموثوقية في ظل ظروف شبكية متنوعة.

نُنشئ مجموعة بيانات جاهزة للتعلم الآلي، تُوثّق أداء البروتوكول عبر أحجام حمولات مختلفة، وفترات إرسال، ومقاييس شبكة مختلفة. يُمكن استخدام هذه المجموعة لتمكين التحليل التنبئي لسلوك البروتوكول، ودعم تصميم أنظمة إنترنت الأشياء القائمة على البيانات.

تُظهر النتائج أن **MQTT** يُحقق كفاءة طاقة وقابلية توسع أفضل للشبكات الكبيرة، بينما يتفوق **CoAP** في سيناريوهات زمن الوصول المنخفض. إلى جانب مقارنة الأداء، يُوفر هذا العمل: (1) إطارًا منهجيًا لتقييم البروتوكول، و(2) مجموعة بيانات مُهيكلة لأبحاث إنترنت الأشياء المستقبلية، تربط تحليل البروتوكول بتطبيقات التعلم الآلي.

تقدم النتائج ومجموعة البيانات المجمعة رؤى عملية لاختيار البروتوكول في البنية التحتية الذكية ونشر إنترنت الأشياء الصناعي، حيث تتطلب الموارد المقيدة حلول اتصال محسّنة.

الكلمات المفتاحية : إنترنت الأشياء، تقييم الأداء ، محاكاة ، التعلم الآلي ، مجموعة بيانات، COOJA، CoAP، MQTT-SN .

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 Addressee Translation

QoS: Quality of Service

TCP: Transmission Control Protocol

UDP: User Datagram Protocol

WSN: Wireless sensor network

XMPP: Extensible Messaging and Presence Protocol

General Introduction

General Introduction:

The rapid expansion of the Internet of Things (IoT), embedding intelligence into billions of resource-constrained devices, has emphasized the urgent need for efficient communication protocols. These devices operate under stringent limitations in processing power, memory, bandwidth, and energy, making protocol selection vital to system feasibility and longevity. Among the application-layer protocols addressing these needs, Message Queuing Telemetry Transport (MQTT) and Constrained Application Protocol (CoAP) stand out. While MQTT uses a publish/subscribe model suited for scalable messaging, CoAP relies on a lightweight request/response architecture ideal for low-latency communication.

MQTT and CoAP serve different use cases and network conditions. However, fragmented and isolated evaluations have made it difficult to determine the most suitable protocol for specific scenarios. A comprehensive, reproducible methodology is required to assess their behavior under constrained conditions and support intelligent protocol selection.

Despite their widespread adoption, no unified framework currently exists to guide the choice between MQTT and CoAP in constrained IoT environments. The lack of structured datasets and standardized simulation procedures inhibits meaningful performance comparisons and limits predictive modeling efforts.

This thesis aims to address this gap by:

- Comparing MQTT and CoAP using controlled simulations in the Cooja network simulator.
- Evaluating key performance metrics: energy consumption, latency, throughput, and packet delivery ratio.
- Exploring the effects of payload size, transmission interval, node density, and network topology.
- Creating a machine learning-ready dataset to support predictive protocol selection.

1. How do MQTT and CoAP differ in performance under constrained network conditions?
2. What impact do environmental parameters have on their efficiency and reliability?
3. Can simulation-generated datasets improve reproducibility and inform intelligent protocol selection?

The evaluation is conducted using the Cooja simulator within the Contiki-NG operating system. Simulations emulate IoT conditions with different number nodes, varying payload sizes, message intervals, and network topologies. Performance metrics are collected and aggregated to build a structured dataset. This dataset is formatted for machine learning tasks, facilitating predictive analyses.

- **Chapter 1**: Literature review of IoT , and IoT application protocols.
- **Chapter 2**: MQTT and CoAP definition.
- **Chapter 3**: Simulation design, Experiments definitions .
- **Chapter 4**: Dataset creation.
- **Chapter 5**: Performance evaluation of MQTT and CoAP. Discussion, insights, and protocol selection guidance.

I. Chapiter 1: Internet of Things

I.1 Introduction

Amidst the contemporary technological revolution, the concept of the Internet of Things (IoT) emerges as a fundamental driver redefining our relationship with the physical environment around us. This chapter aims to build a comprehensive and solid foundation for the entire study. We will begin by providing a detailed introduction to the world of the Internet of Things, in terms of its definition, architecture, and practical applications. We will then move directly to exploring the "language" of these devices—the communication protocols that enable them to exchange data effectively. We will conduct a comprehensive survey of the most prominent protocols operating at the application layer, analyzing their characteristics and operating models, providing the reader with a comprehensive understanding of the general framework and available options before focusing on the specific protocols in our study.

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.1 Internet of things

I.2.1 The Internet of Things Definition

- **Definition 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.[02]

- **Definition 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." [03]

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. [04]

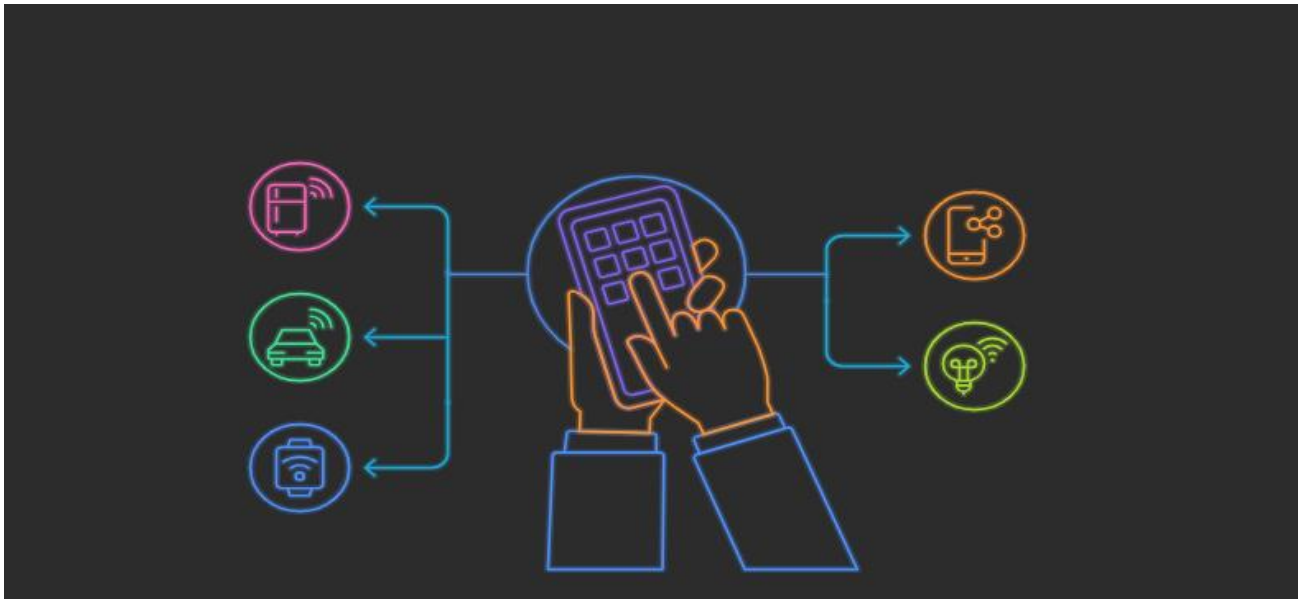


Figure 1:Internet of Things [05]

I.2 Applications of the Internet of Things

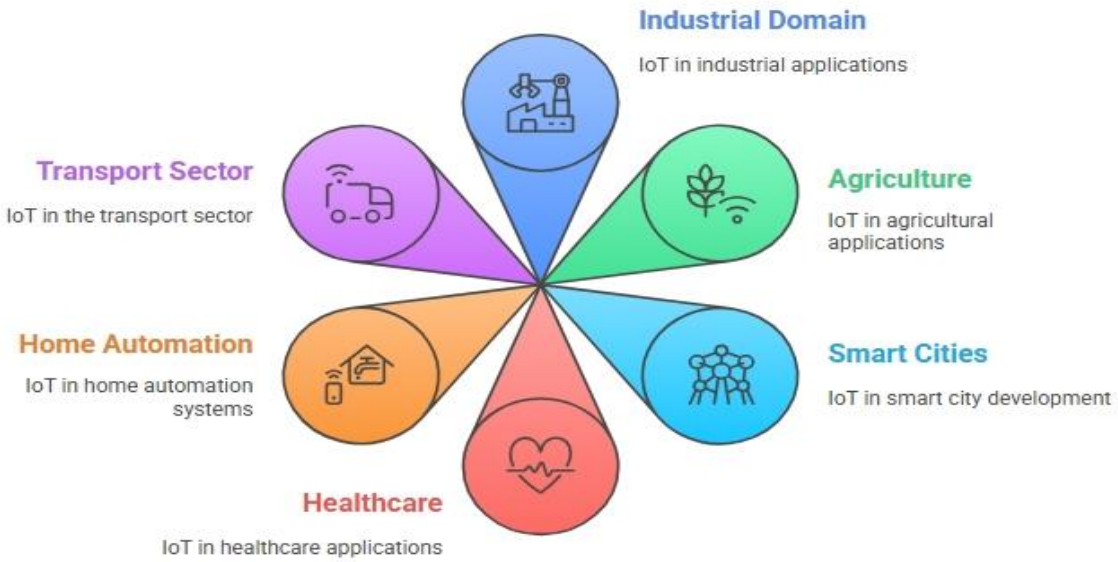


Figure 2:IOT Application area[05]

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. [06].

- **Agriculture:** The Internet of Things (IoT) can be utilized to monitor crop environments through interconnected sensor networks [01]. 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 [01]. 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 [06]. 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. [07]

I.3 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. [08]

- **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. [09]

- **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. [01]

- **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. [08]

- **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. [08]

- **Operating applications:** Operating applications are human-machine interfaces (HMI) that allow users to view data in the form of dashboards. [08]

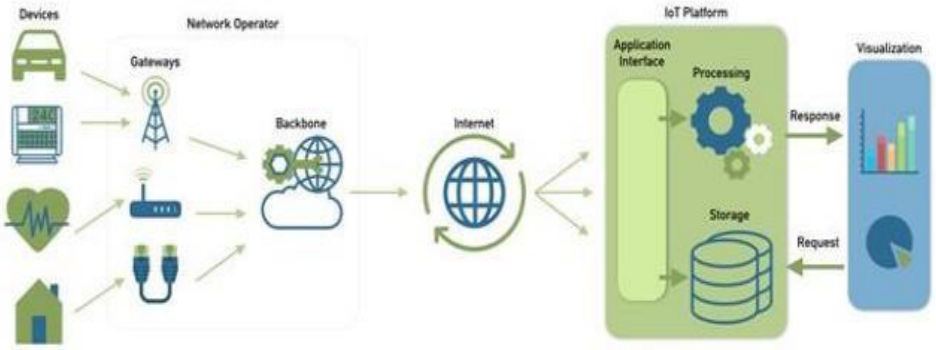


Figure 3:How The Internet of Things Works [10]

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. [08]

I.4.3 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. [13]

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. [13]

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. [13]

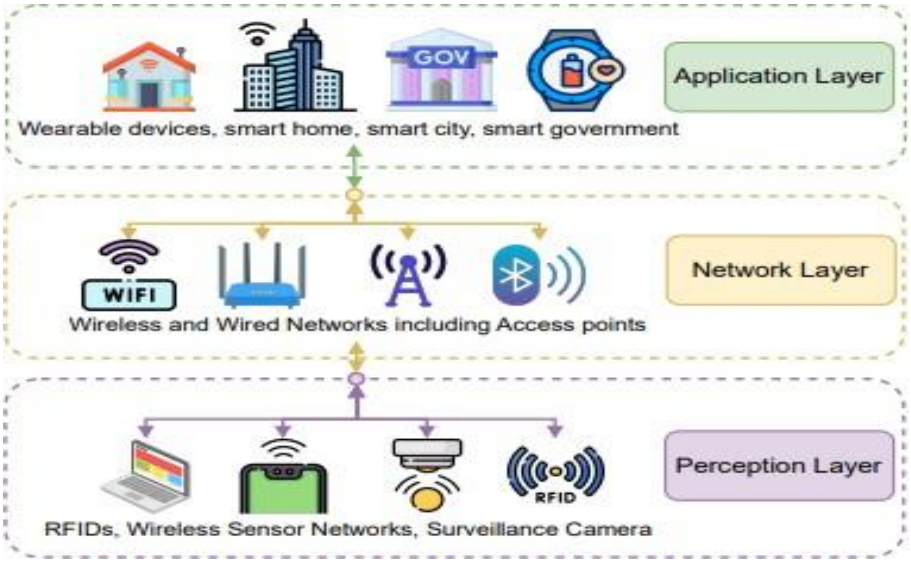


Figure 4: three-layer architecture [14]

B. The five-layer architecture:

The architecture in question is made up of five layers

:

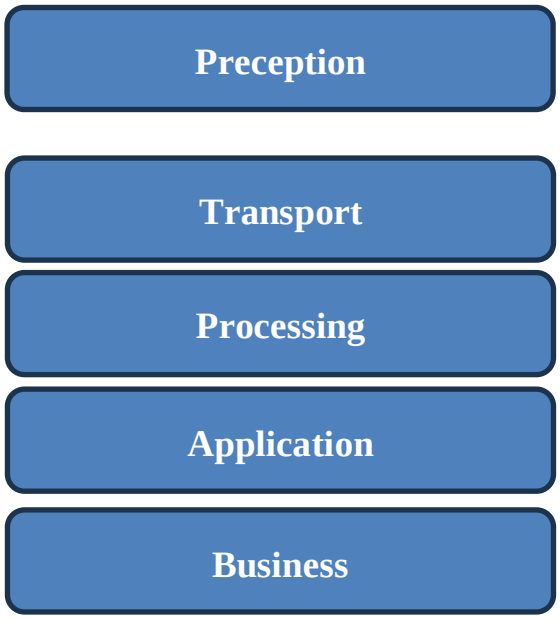


Figure 5: Five layer iot architecture

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. [13]

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. [13]

The business layer of the Internet of Things (IoT) is responsible for managing the entire system, including applications, business models, and user privacy issues. [13]

I.4 The protocols of 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.

Tableau 1:The protocols of IoT

Application layer	COAP.HTTP.EBHTTP.LTP.SNMP.IPFIX.DNS.NTP.SSH.DLMS. COSEM.DNP.MODBUS
Network layer	IPV6/IPV4.RPL.TCP/UDP.ULP.SLIP.6LOWPAN
Preception layer	IEEE802.11Series,802.15Series,802.3,802.16, WirelessHART ,Z-WAVE, UWB ,IRDA ,PLC,LonWorks, KNX

I.5 Application in IOT:

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. [16]

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. [17]

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. [16]

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. [17]

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. [17]

I.6 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).

I.6.1 Application layer protocols

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.

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.

Tableau 2:Application layer protocols

Application	Transport	Network	Data Binding	Physical
COAP	UDP	6LoWPAN	LPWAN	Ethernet
MQTT	TCP	RPL	MQTT Binding (custom/simple)	NFC
AMQP	TCP	IP (IPv6, IPv4, ...)	IEEE 802.15.4	BLE
DDS	UDP	ICMP	RTPS (Real-Time Publish-Subscribe)	LTE
XMPP	TCP	IP (IPv4/IPv6)	XML Binding	RFID
API REST	TCP	IP (IPv4/IPv6)	HTTP Binding	Wi-Fi / 802.11
WebSocket	TCP	IP (IPv4/IPv6)	JSON / WebSocket Binding	ZigBee Z-Wave

A. CoAP (Constrained Application Protocol)

The Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained networks in the Internet of Things. The protocol is designed for machine-to-machine (M2M) applications such as smart energy and building automation. [23]

• Operation:

CoAP operates on a client/server request/response model. It utilizes two fundamental message types: Confirmable (CON), which requires an acknowledgment to ensure reliable delivery, and Non-confirmable (NON), which is sent without the need for an acknowledgment for less critical data.[23]

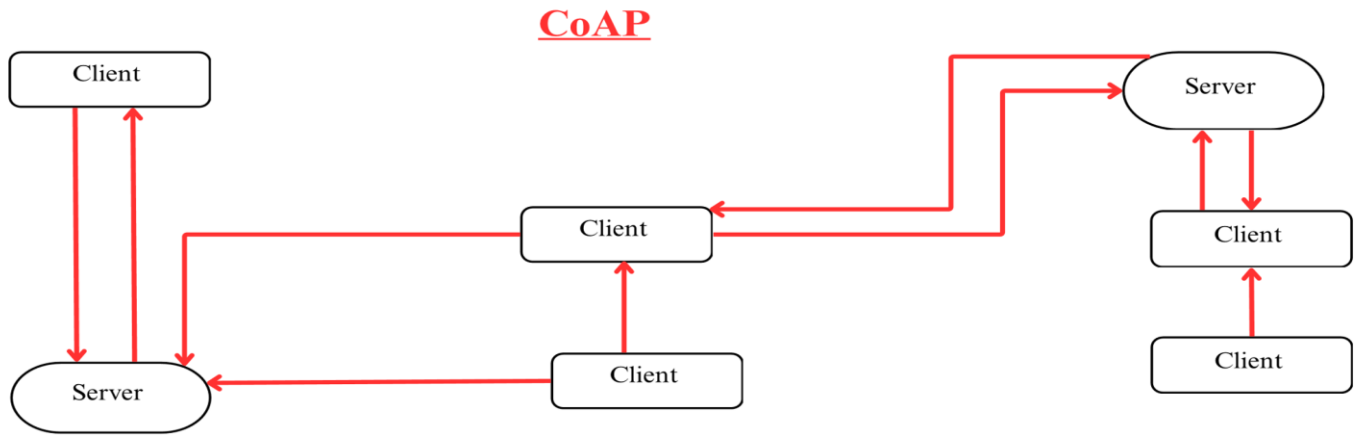


Figure 6: Operation protocol CoAP .[05]

B. MQTT (Message Queuing Telemetry Transport)

MQTT is a client/server publish/subscribe messaging transport protocol. It is designed to be lightweight and open, making it suitable for connecting constrained devices in scenarios with high-latency or limited-bandwidth networks, providing a reliable messaging service.[24]

• Operation:

Its operation is centered on a message broker. Clients connect to the broker and can publish messages to specific "topics" or subscribe to topics to receive messages. The broker is responsible for receiving messages from publishers and forwarding them to the relevant subscribers.[24]

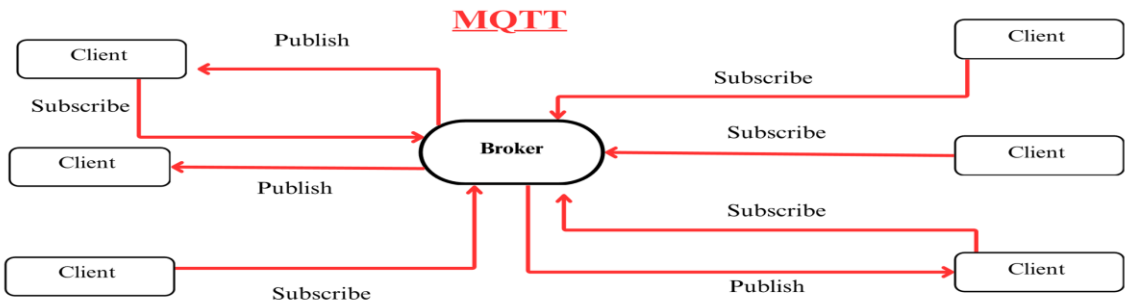


Figure 7:Operation Protocol MQTT [05]

C. 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. [18]

● 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." [05]

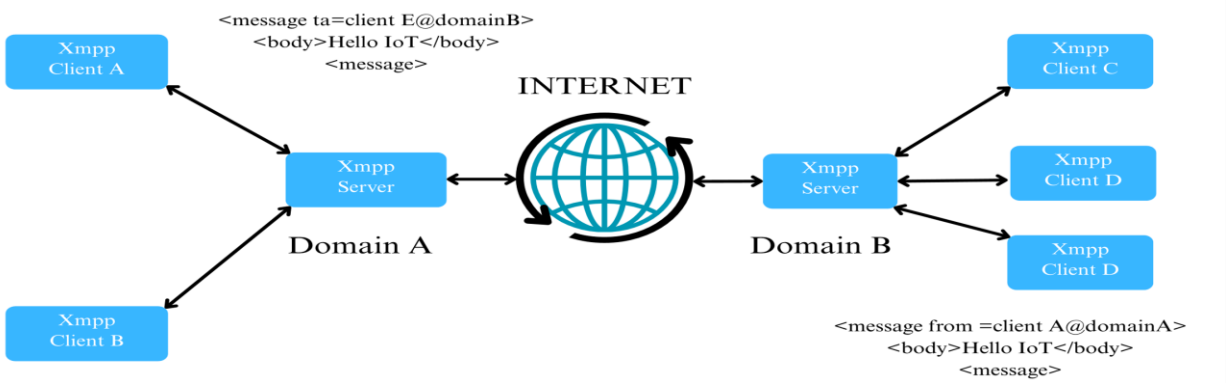


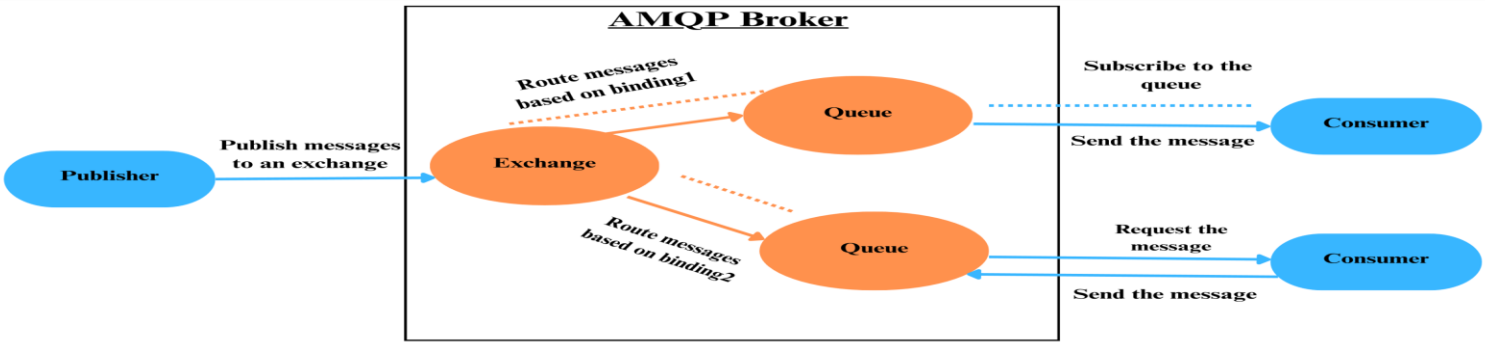
Figure 8: Operation Protocol XMPP[05]

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. [19]

● 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. [05]



Figur 9 : Operation protocol AMQP

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. [18]

● 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. [05]

Figure 10:Operation protocol WebSocket . [05]

F. 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. [18]

● 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. [05]

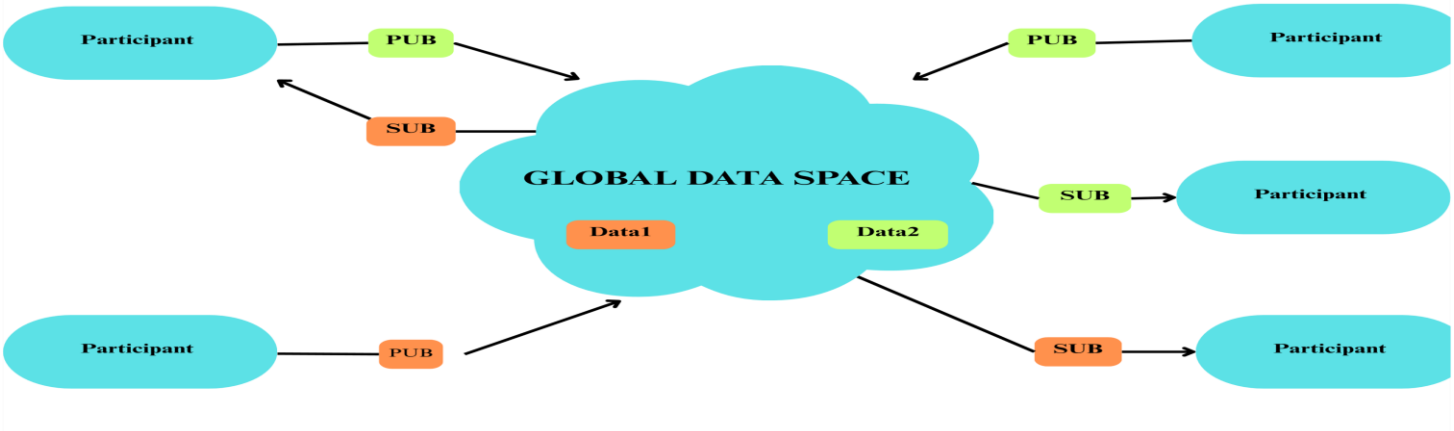


Figure 4:Operation protocol DDS. [05]

G.HTTP (Hyper Text Transport Protocol):

HTTP is primarily a web message technology created by Tim Berners-Lee. It was later developed by the IETF and the W3C and initially published as a standard protocol [11]. RESTful Web architecture is supported by HTTP request/response. HTTP, like CoAP, utilizes URIs, The URI is used by the server to deliver data and the URI is used by the client to receive data.

HTTP is a text-based protocol that leaves the size of headers and message payloads to the web server or programming language. TCP is the default transport protocol for HTTP, and TLS/SSL is used for security. As a result, client-server communication is connection-oriented. Because http does not define QoS clearly, HTTP is a widely used web messaging protocol that includes capabilities such as permanent connections, request pipelining, and chunked transfer encoding [5], [6], [13].

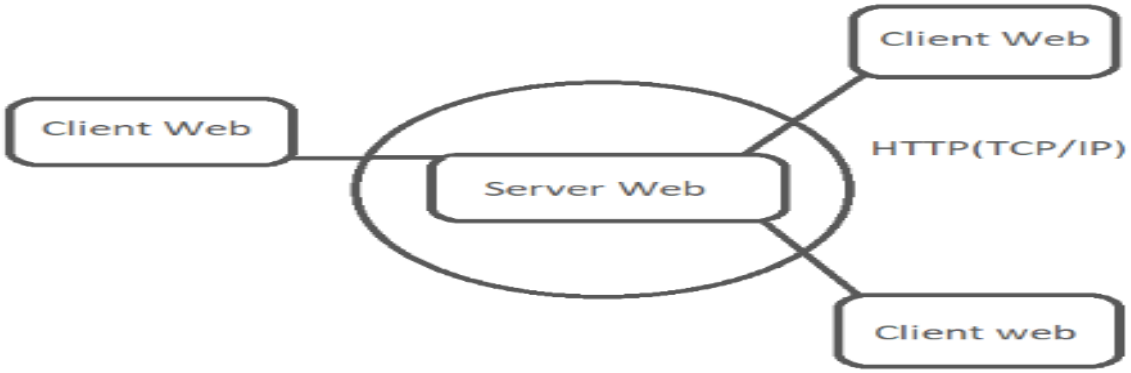


Figure 5: HTTP protocol

I.9 Comparative Analysis of IOT 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.

Tableau 3:Comparison of different application layer protocol[15] .

	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
Framework	Jabber, XMPP Framework	Emqtt, HiveMQ, Mosquitto, Eclipse Paho	Eclipse Californium, nCoA	RabbitMQ, Storm	Jetty websocket, Apache Tomcat	
Protocol type	Messaging	Messaging	Messaging	Web transfer	Messaging	Messaging

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.

Conclusion

In this chapter, we covered the foundational principles of the Internet of Things and extensively reviewed the most important communication protocols at the application layer. From this overview, it becomes clear that MQTT and CoAP are particularly important due to their design for resource-constrained environments, making them ideal candidates for an in-depth comparative study. Now that we have a grasp of the general framework, the next step requires us to move from a comprehensive overview to a more detailed analysis. Therefore, in the next chapter, we will focus our lens on these two protocols in particular, delving into their technical details to uncover their inner workings.

II. Chapiter 2 : MQTT-SN and CoAP presentation

II . 1 Introduction

Building on the findings of the previous chapter, this chapter devotes its efforts to providing an in-depth technical analysis of the two selected protocols, MQTT-SN and CoAP. Having provided an overview of a set of protocols, it is now time to dissect these two protocols in particular. In this chapter, we will explore the architecture, operating model, message structure, and reliability and security mechanisms of each. The goal is to build a solid technical understanding that will later allow us to interpret the performance results we obtain in a test environment.

II.2 MQTT Protocol:

II.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.[19]

II.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)

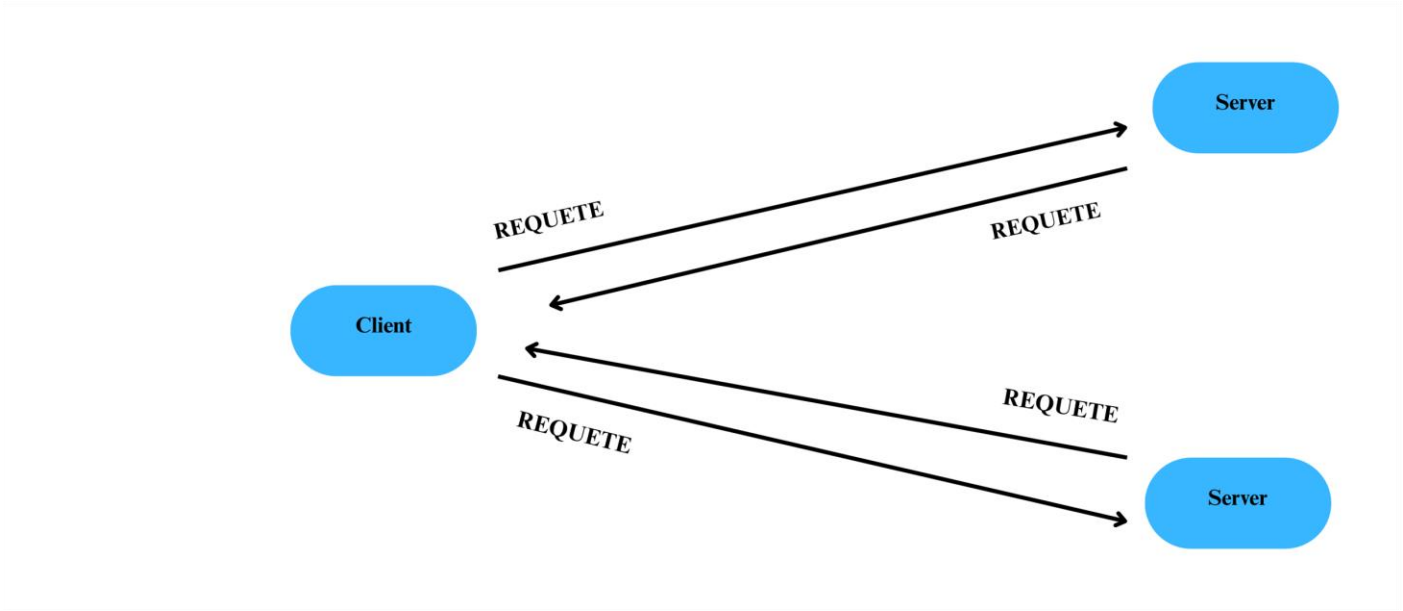


Figure 6 : Client/Server Principale

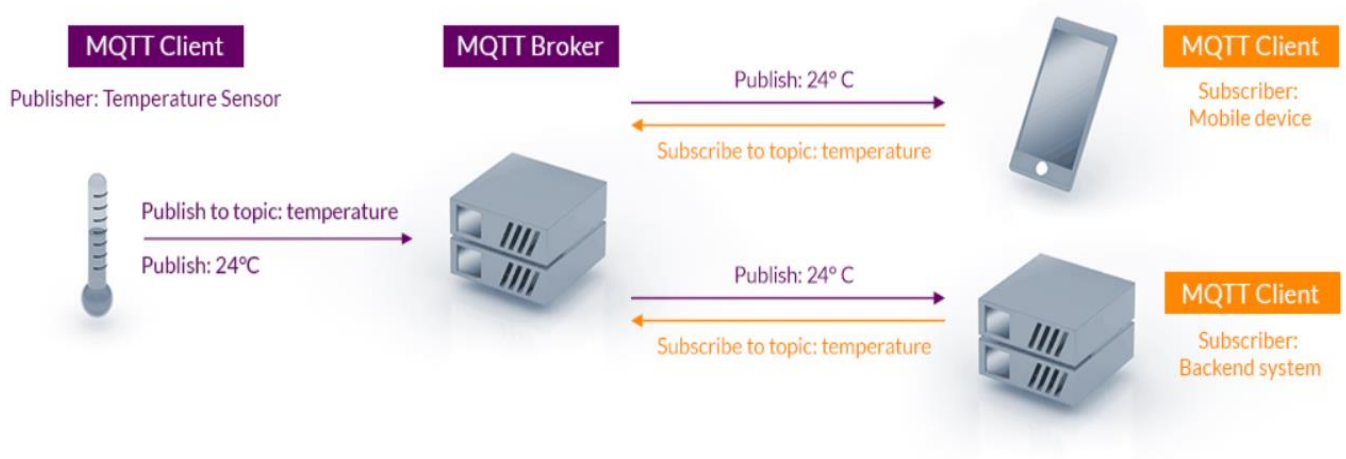


Figure 7: Principle of Operation of the MQTT Protocol . [20]

II.2.3 MQTT Architect

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

a. 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

II.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.

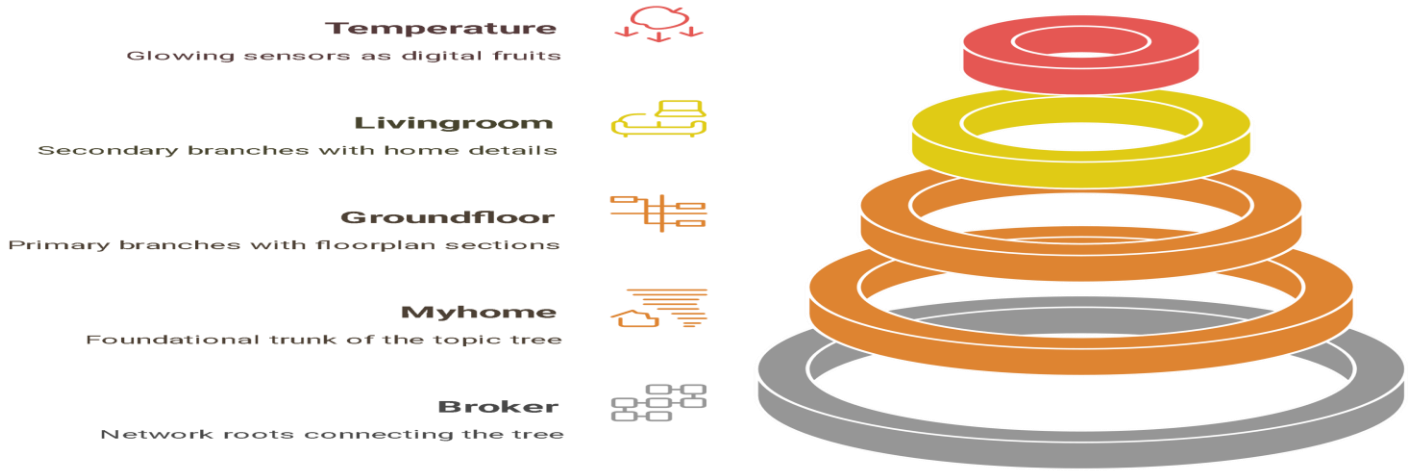


Figure 8:Topic Level Separator . [21]

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

Tableau 4: Structure of an MQTT Control Packet. [21

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

Fixed header :

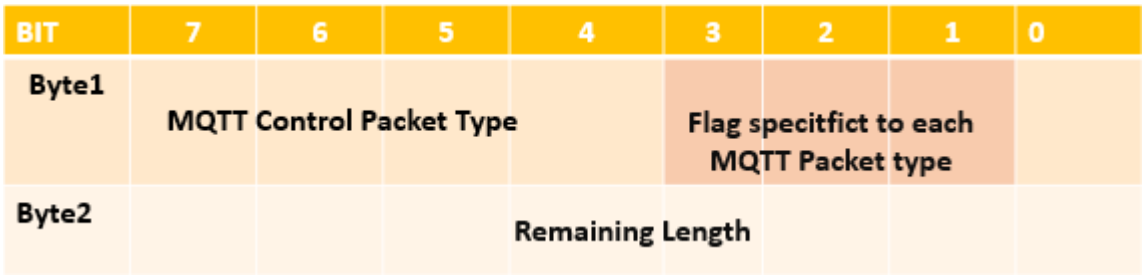


Figure 9 : Fixed header Format. [21]

- **The components of the header:**

A. MQTT Control Pocket Types

Tableau 5: Type of MQTT Packets . [21]

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 acknowledgment (QoS 1)
PUBREC	5	Client to Server or Server to Client	Publish received (QoS 2 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

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 5 below.

DUP = Duplicate delivery of a PUBLISH packet.
QoS = PUBLISH Quality of Service.
RETAIN = PUBLISH retained message flag.

Tableau 6: Description of the flag fixed header of the MQTT protocol . [21]

MQTT Control Packet	Fixed Header flags	Bit 3	Bit 2	Bit 1	Bit 0
CONNECT	Reserved	0	0	0	0
CONNACK	Reserved	0	0	0	0
PUBLISH	Used in MQTT v5.0	DUP	QoS		RETAIN
PUBACK	Reserved	0	0	0	0
PUBREC	Reserved	0	0	0	0
PUBREL	Reserved	0	0	1	0
PUBCOMP	Reserved	0	0	0	0
SUBSCRIBE	Reserved	0	0	1	0
SUBACK	Reserved	0	0	0	0
UNSUBSCRIBE	Reserved	0	0	1	0
UNSUBACK	Reserved	0	0	0	0
PINGREQ	Reserved	0	0	0	0
PINGRESP	Reserved	0	0	0	0
DISCONNECT	Reserved	0	0	0	0
AUTH	Reserved	0	0	0	0

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. [21]

II.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.

II.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.

II.2.7 CONNECT:

The following table 6 is an example of a CONNECT packet:

Tableau 7: Format of a CONNECT Packet . [21]

	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

Description of the CONNECT packet fields:

Tableau 8: CONNECT Packet Fields. [21]

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.**

II.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). [21]

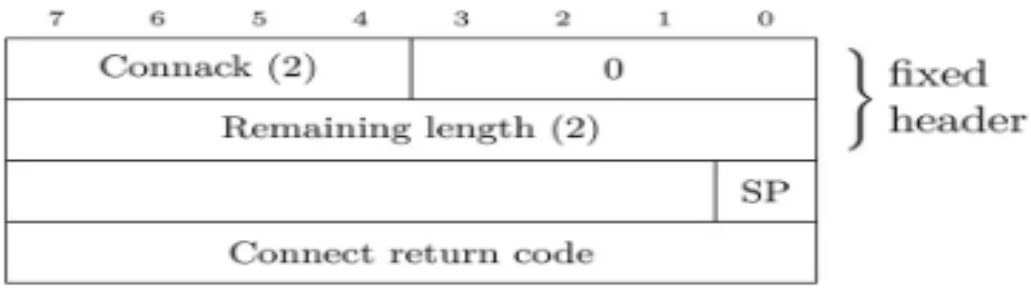


Figure 10: Format Packet CONNACK. [22]

II.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.

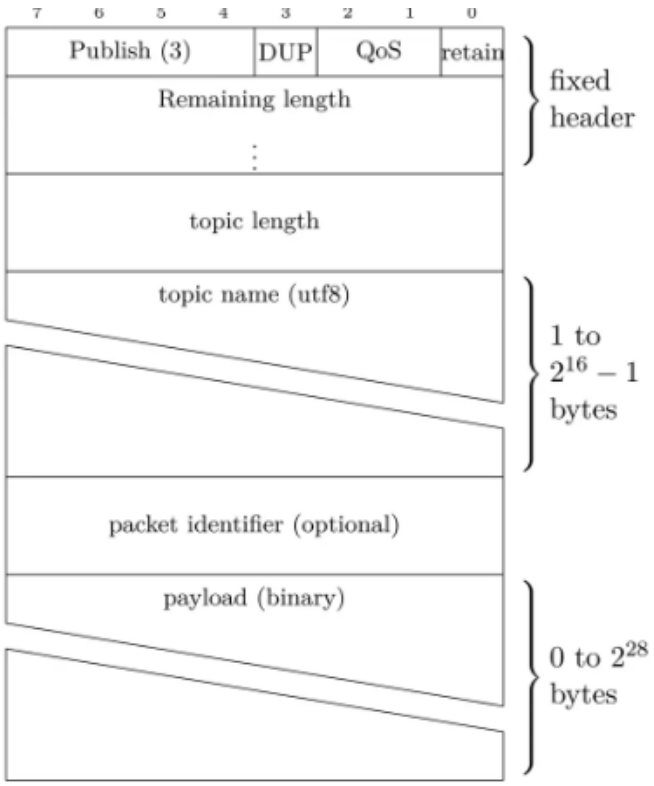


Figure 11 :Format of a PUBLISH Packet:[22] .

II.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.

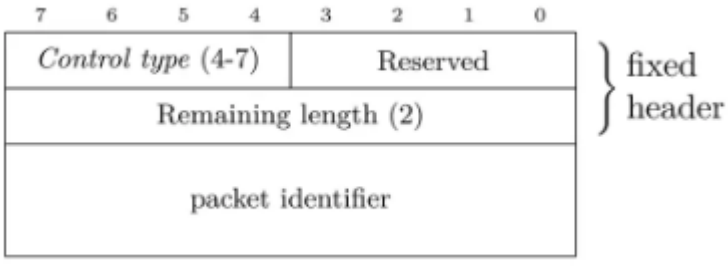


Figure 12 :Format of a PUB[ACK/REC/REL/COMP] packet. [22]:

II.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.

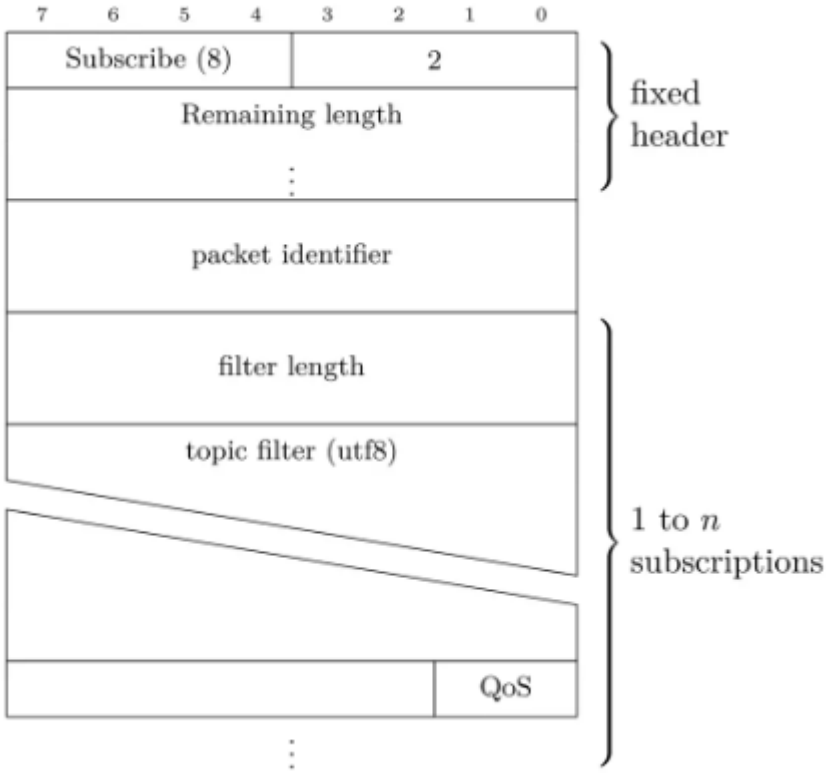


Figure 20 :Format of a SUBSCRIBE packet. [22]

II.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. [21]

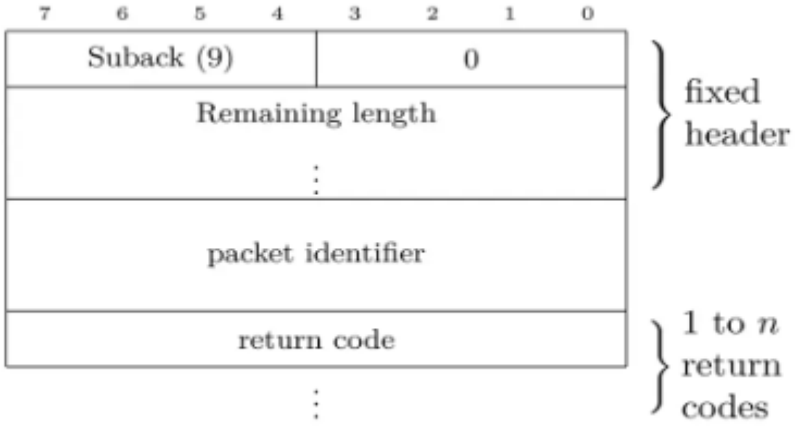


Figure 21:Format of a SUBACK packet.[22]

II.2.13 UNSUBSCRIBE:

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

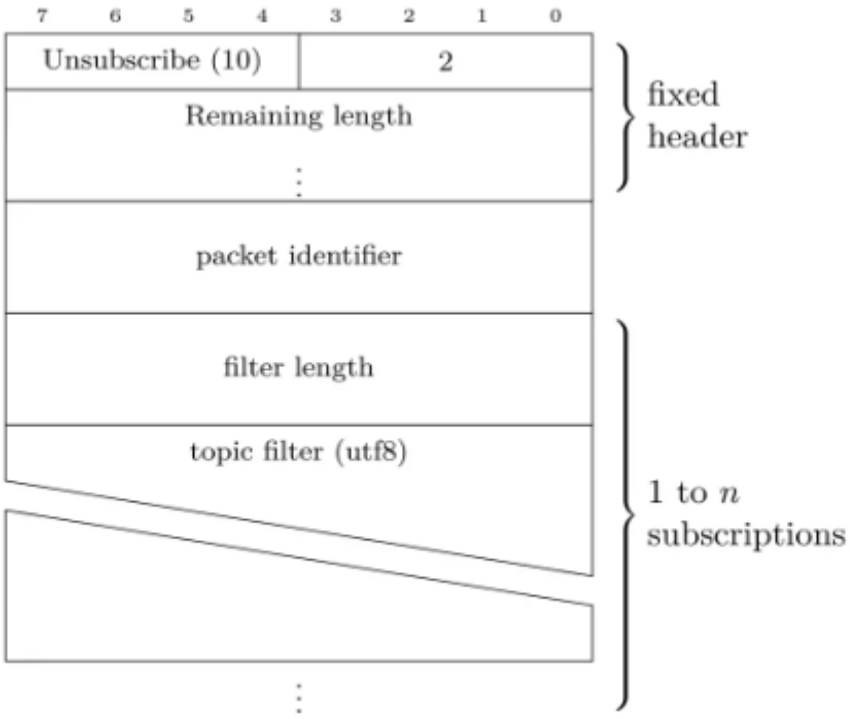


Figure 22 :Format of an UNSUBSCRIBE packet[22]

II.2.14 UNSUBACK

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

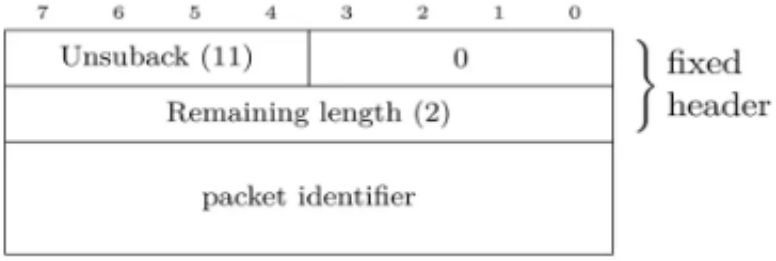


Figure 23 : Format of an UNSUBACK packet . [22]:

II.2.15 PINGREQ, PINGRESP

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

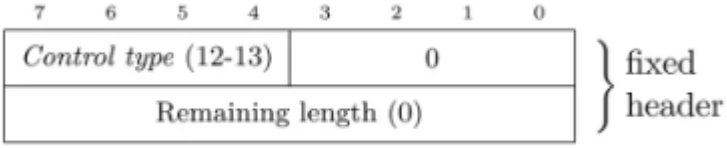


Figure 24 :Format of a PING packet [REQ/RESP]. [22]

II.2.16 DISCONNECTION

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

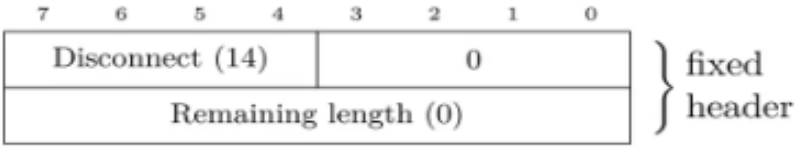


Figure 25 :Format of a DISCONNECT packet. [22]

II.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 files sample possible.

II.3 The CoAP protocol

II.3.1 Definition of CoAP

The Constrained Application Protocol (CoAP) is a specialized web transfer protocol defined by the IETF CoRE working group, specifically designed for use with constrained nodes and networks typical in the Internet of Things. It follows a RESTful (Representational State Transfer) architecture, enabling devices to interact with well-defined resources using methods like GET, POST, PUT, and DELETE, similar to HTTP.

CoAP is engineered to facilitate machine-to-machine (M2M) interactions and easily integrate with the existing web through proxies, while addressing the specific needs of constrained environments. Key design objectives include very low overhead, simplicity to avoid complex parsing, and support for asynchronous communication, which is vital for energy-saving, sleepy devices that cannot be responsive at all times.

The protocol operates over UDP to reduce overhead and complexity, though it also has specifications for use over other transports like TCP and DTLS for security. Each piece of information is treated as a resource identified by a Universal Resource Identifier (URI), allowing the protocol to be a fundamental building block for a web of things, extending the familiar web paradigm to highly constrained devices. [23]

II.3.2 Features of CoAP: (CONSTRAINED APPLICATION PROTOCOL):[28]

1. Web protocol fulfilling M2M requirements in constrained environments.
2. User Datagram Protocol (UDP) binding with the optional reliability, supporting unicast and multicast requests.
3. Security binding to Datagram Transport Layer Security (DTLS).
4. A stateless HTTP mapping, allowing proxies to be built providing access to CoAP resources via HTTP in a uniform way or for HTTP simple interface to be realized alternatively over CoAP.
5. Simple proxy and caching capabilities.

6. URI and Content-type support
7. Low header overhead and parsing complexity.
8. Asynchronous message exchanges.

II. 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 13:Principle of Operation of The COAP protocol

II .3.3.1 COAP architecture:

The most important aspect of the CoAP protocol structure is that it avoids message fragmentation, allowing CoAP packets to fit into a single Ethernet or IEEE 802.15.4 frame. The CoAP Message Layer is designed to handle UDP and asynchronous switching, as seen in "Figure", while the request/response layer controls the communication mechanism.

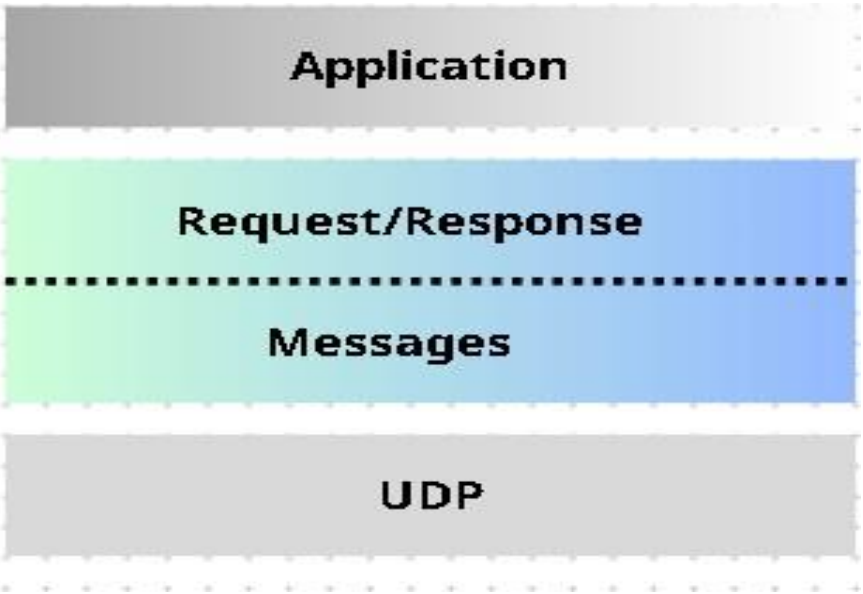


Figure 14: COAP Architecture

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:

Tableau 9: COAP Method

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

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 in:

Tableau 10:COAP Response Codes

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

❖ 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.

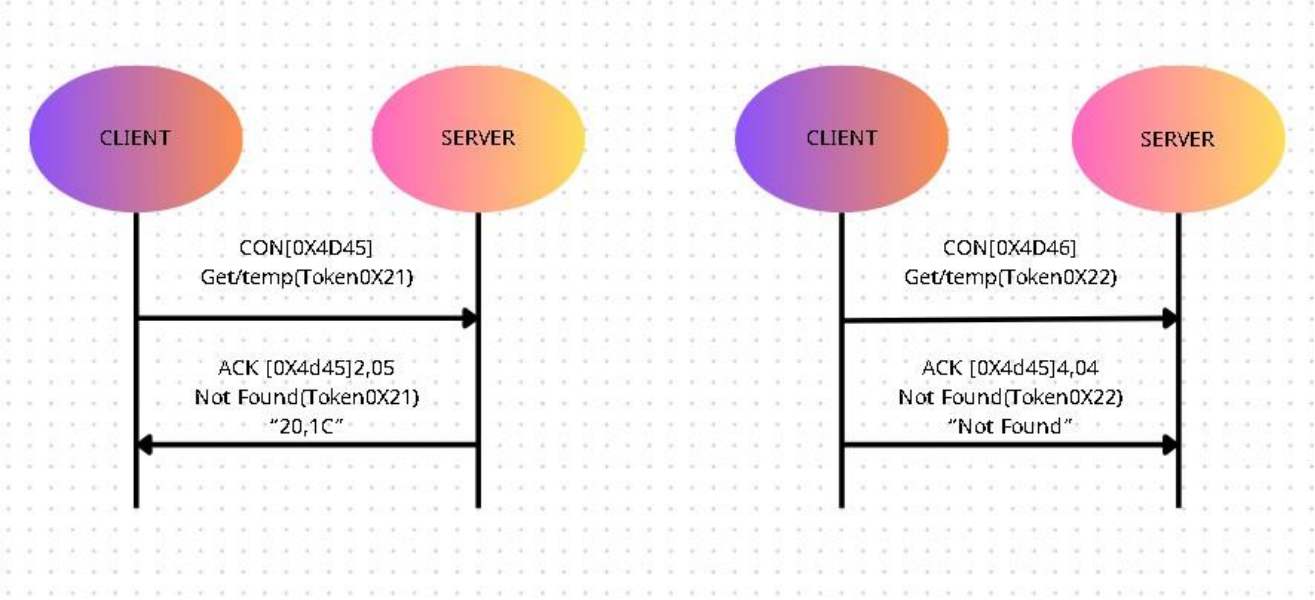


Figure 15: The Successful and Failure Response Results of GET Method

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).

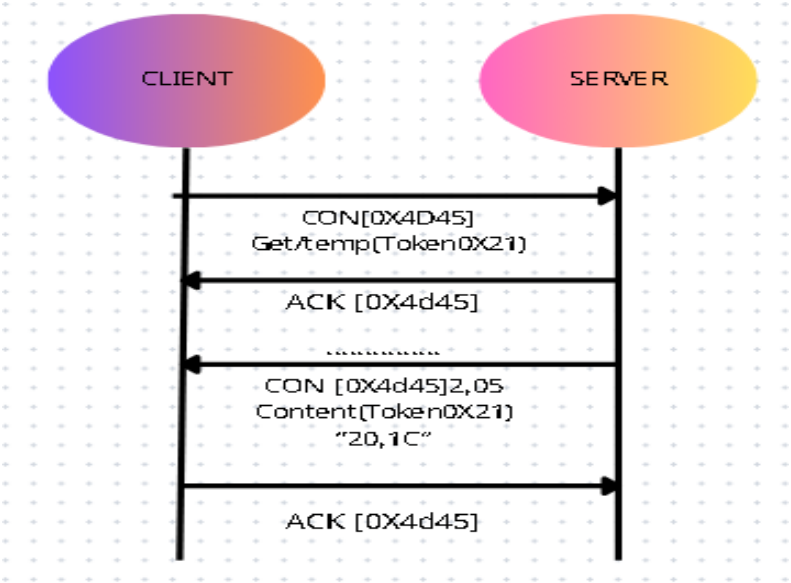


Figure 16:Get Request with a Separate Response

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).

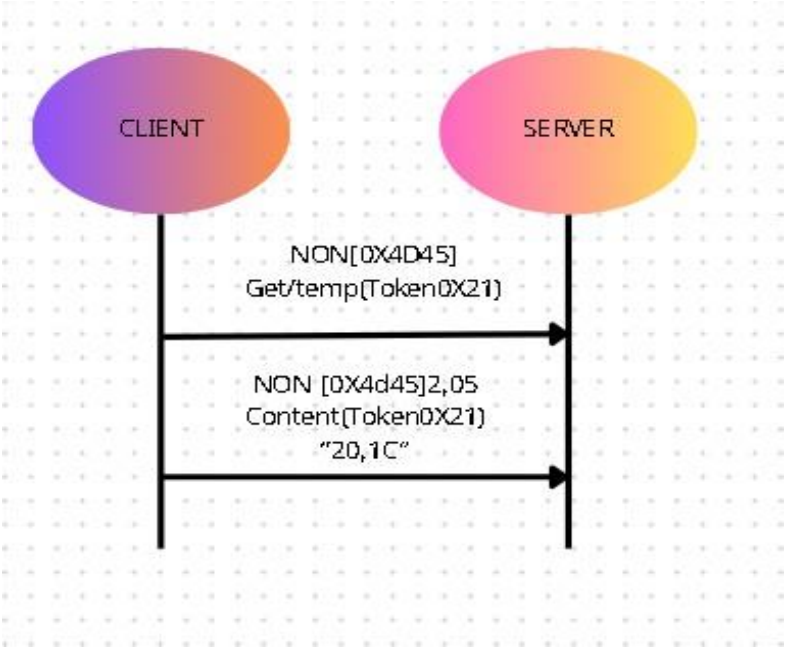


Figure 30:Non Confirmablr Request and Response

II.3.4 Message CoAP

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

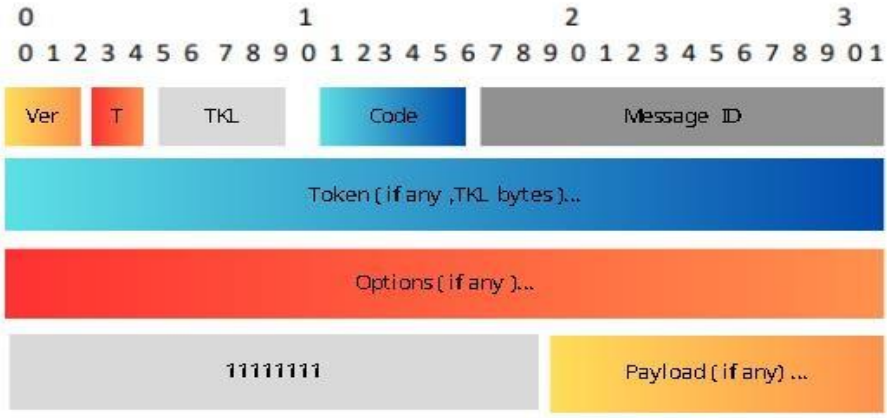


Figure 31:FORMAT du Message COAP.[26]

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), Non-confirmable (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.

II. 4 Comparison

II. 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.

Tableau 11:Major differences between MQTT and COAP[26]

	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 model	Request/Response	Publication/Subscription
Header	4 Bytes	2 Bytes
Messaging mode	Uses both asynchronous and synchronous mode.	Uses asynchronous mode only
Number of message types used	4	16

II.4.2 Conclusion of comparison

After presenting the CoAP protocol and the MQTT protocol and explaining how each works, we can say that both protocols are useful as IoT protocols, but with fundamental differences.

The MQTT protocol has an architecture based on publish/subscribe and broker in the middle, which makes it ideal for communication over a wide-area network. This protocol is also useful in cases where bandwidth is limited, which is not bad.

For the CoAP protocol, what differentiates it from MQTT and makes it stronger is its compatibility with HTTP, it is built on the UDP protocol and this is very useful for using CoAP in certain resource-limited environments, because we mustn't forget that UDP allows broadcast and multicast, so transmission can be made to several hosts using less bandwidth, which makes this protocol ideal for local network environments where devices need to talk to each other quickly.

II. 5 Conclusion

Having dissected the technical and operational aspects of both MQTT-SN and CoAP in this chapter, we now have a clear theoretical view of their potential strengths and weaknesses. We understand how each works on paper. However, theoretical evaluation remains insufficient to judge their effectiveness in practical scenarios. Therefore, it is now necessary to move from the theoretical stage to practical testing. In the next chapter, we will put these protocols under the experimental microscope, detailing the simulation methodology we used and the tools and scenarios designed to evaluate their actual performance.

III Chapiter 3 :
Simulation of the protocols

III.1 Introduction

Having completed the theoretical foundation in the previous chapters, we now move to the heart of the practical aspect of this study. This chapter focuses exclusively on the simulation methodology adopted to evaluate the performance of the MQTT-SN and CoAP protocols. We will begin by justifying the choice of simulation as the evaluation method, then detail the tools used, most notably the COOJA simulator and the Contiki operating system. We will then describe the experimental environment in detail, identify the parameters that were modified, and design various scenarios aimed at testing the two protocols under various network conditions and realistic simulations.

III.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 :

III.3 Analytical Methods

III.3.1 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. . . [27]

III.3.2 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.[28]

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.

III.4 TOOLS USED IN THIS SIMULATION:

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

III.4.1 Software

Contiki OS

Contiki is a lightweight and flexible open-source operating system designed for IoT nodes. It was created by Adam Dunkel's 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. .[29]

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 high- level 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 [29].

III.4.2 Programming Languages Usage

Python language:

Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with its notable use of significant indentation. Created by Guido van Rossum and first released in 1991, Python's dynamic typing and garbage-collected memory management, combined with its support for multiple programming paradigms—including structured, object-oriented, and functional programming—make it a versatile tool for a wide range of applications.

A key strength of Python is its "batteries-included" philosophy, which is reflected in its comprehensive standard library that provides tools suited to many tasks. This is further extended by a vast ecosystem of third-party packages, particularly in the domain of scientific computing and data analysis. Libraries such as NumPy, Pandas, and Matplotlib have established Python as a leading platform for data science, enabling complex numerical computation, sophisticated data manipulation, and high-quality data visualization. [25]

This is part of the Python code we used to convert the simulation output into data that is easy to read and analyze.

```
    base_data_original_typed = processed_initial_data_typed.get(BASE_SCENARIO_KEY)
    if not base_data_original_typed:
        print(f" لم يتم العثور على بيانات السيناريو الأساسي '{BASE_SCENARIO_KEY}' .")
        exit()

    cpu_t, lpm_t, tx_t, rx_t = estimate_ticks_from_energy(base_data_original_typed["استهلاك الطاقة الكلي (J)"], base_data_original_typed)
    base_data_original_typed["طاقة CPU (ticks)"] = cpu_t
    base_data_original_typed["طاقة LPM (ticks)"] = lpm_t
    base_data_original_typed["طاقة TX (ticks)"] = tx_t
    base_data_original_typed["طاقة RX (ticks)"] = rx_t
    base_data_original_typed = recalculate_dependent_metrics(base_data_original_typed, DEFAULT_PAYLOAD_SIZE)

    print(f"القيم الأساسية المرجعية لـ {BASE_SCENARIO_KEY} (التجربة الأولى):")
    for k, v in base_data_original_typed.items(): print(f"  {k}: {v}")

    for scenario_tuple in final_coap_scenarios_to_generate:
        scenario_key_name, s_clients, s_servers, t_trans_val, s_payload_val = scenario_tuple
        print(f"جاري معالجة السيناريو: {scenario_key_name} (C:{s_clients}, S:{s_servers}, T:{t_trans_val}, P:{s_payload_val})...")

        estimated_first_run_data_dict = {}
        temp_custom_vals = {}
        payload_for_calc = DEFAULT_PAYLOAD_SIZE

        if scenario_key_name == BASE_SCENARIO_KEY:
            estimated_first_run_data_dict = copy.deepcopy(base_data_original_typed)
        elif s_clients != DEFAULT_CLIENTS and s_servers == DEFAULT_SERVERS and t_trans_val == DEFAULT_TRANS_INTERVAL and s_payload_val == DEFAULT_PAYLOAD_SIZE:
            if s_clients in CLIENT_SCENARIO_CUSTOM_VALUES: temp_custom_vals = CLIENT_SCENARIO_CUSTOM_VALUES[s_clients]
        elif s_servers != DEFAULT_SERVERS and s_clients == DEFAULT_CLIENTS and t_trans_val == DEFAULT_TRANS_INTERVAL and s_payload_val == DEFAULT_PAYLOAD_SIZE:
            if s_servers in SERVER_SCENARIO_CUSTOM_VALUES: temp_custom_vals = SERVER_SCENARIO_CUSTOM_VALUES[s_servers]
        elif t_trans_val != DEFAULT_TRANS_INTERVAL and s_clients == DEFAULT_CLIENTS and s_servers == DEFAULT_SERVERS and s_payload_val == DEFAULT_PAYLOAD_SIZE:
            if t_trans_val in TRANS_INTERVAL_SCENARIO_CUSTOM_VALUES: temp_custom_vals = TRANS_INTERVAL_SCENARIO_CUSTOM_VALUES[t_trans_val]
        elif s_payload_val != DEFAULT_PAYLOAD_SIZE and s_clients == DEFAULT_CLIENTS and s_servers == DEFAULT_SERVERS and t_trans_val == DEFAULT_TRANS_INTERVAL:
            if s_payload_val in PAYLOAD_SIZE_SCENARIO_CUSTOM_VALUES:
                temp_custom_vals = PAYLOAD_SIZE_SCENARIO_CUSTOM_VALUES[s_payload_val]
                payload_for_calc = s_payload_val
```

C language:

C is a general-purpose, procedural programming language developed by Dennis Ritchie at Bell Labs in the early 1970s. It was originally designed for system programming, particularly for writing operating systems (like UNIX).

III.4.3 Hardware

- PC 1

- ✓ We used a DELL laptop
- ✓ RAM: 4.00 GB.
- ✓ System: Windows 10 64-bit
- ✓ Processor: i3 4G

- PC 2

- ✓ We used a DELL laptop
- ✓ RAM: 32.00 GB.
- ✓ System: Ubuntu 20.04
- ✓ Processor: Intel Core i9-12900H

III. 5 The stages of the simulation

Installing the Contiki operating system was the easiest phase:

III.5.1 Running the Cooja simulator:

The simple way to run Cooja is to run it in its own directory figure (32) cd contiki/tools/cooja ant run

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

Figure 32:Simulation Execution

After running Cooja, the following window appears:

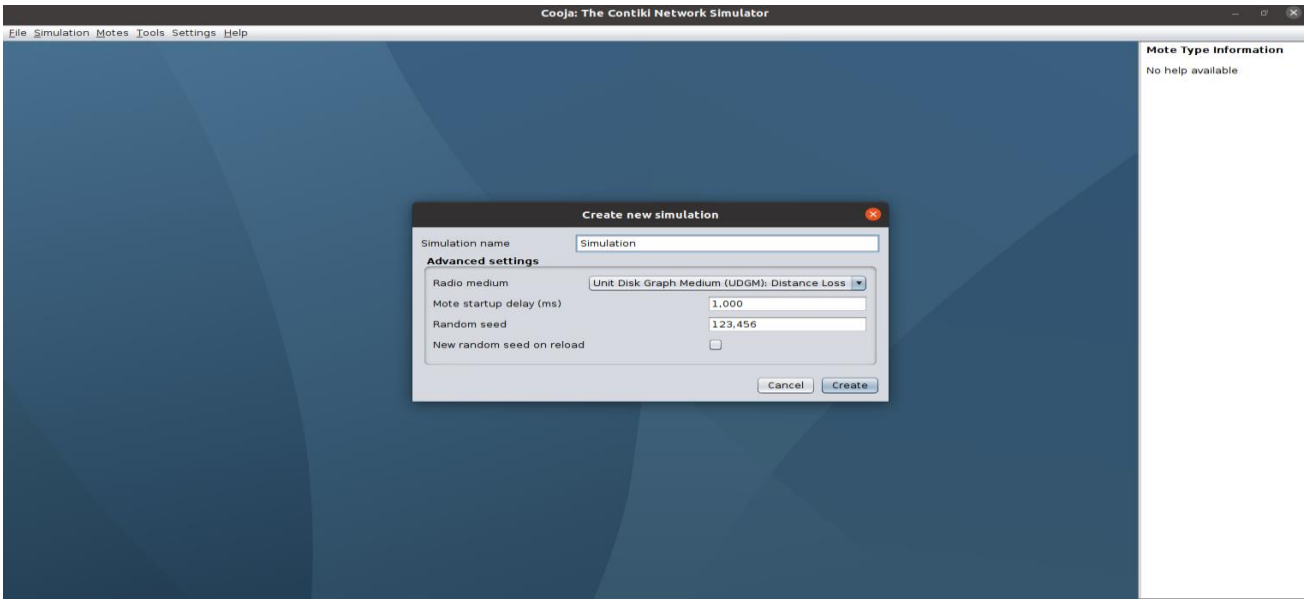


Figure 33:COOJA Simulator Interface

III.5.2 Creation of a new simulation

In the menu, you must choose: File > New simulation (Figure 33). 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 .

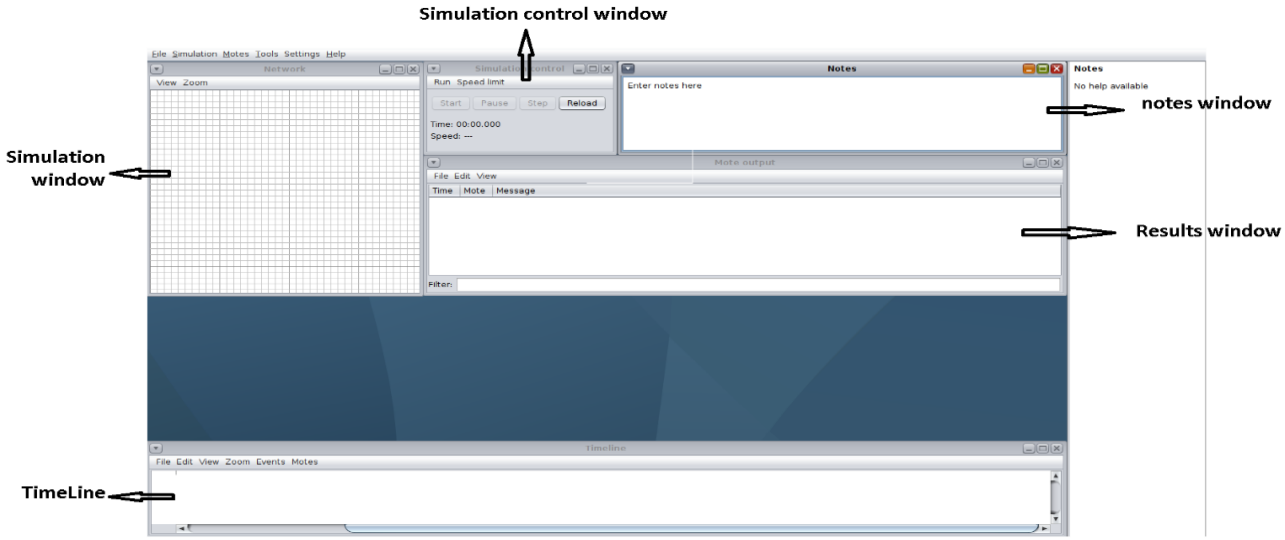


Figure 34:COOJA Simulator Window

III.6 Simulation Environment

The simulation presents the major parameters of the environment. The study focuses on imitating protocols within a network containing 10 servers. Initially, the nodes are randomly distributed within an area of 100 × 100 m. The nodes run at a speed of 1 meter per second. The maximum speed and punctuation time defines the dynamics pattern of the model. The break time is set for 5 seconds. Various landscapes will be imitated, in which each simulation will last for one hour. The default values for some environmental parameters are wide in the table below.

Tableau 12: Simulation Environment

Parameters	Value
Surface	100x100 m
Phase initialization	1 minute
Time simulation	30 minutes
Random seed	123,456
Mote startup delay	5s
Speed of nodes	1s
Break time	2s
Number of publisher	10 - 20 - 30 - 40 - 50 - 70 - 100
Number of subscriber	10 - 20 - 30 - 40 - 50 - 70 - 100
Number of servers	10 - 20 - 30 - 40 - 50 - 60 - 70 - 100
Number of clients	10 - 20 - 30 - 40 - 50 - 60 - 70 - 100

III.6.1 Scenario of the CoAP protocol Simulation

Once the new simulation is created and named, we must create the motes we have chosen “Z1 Mote” as type motes, we have choose for the server as quantity (10 server), the client (10 client) and one router

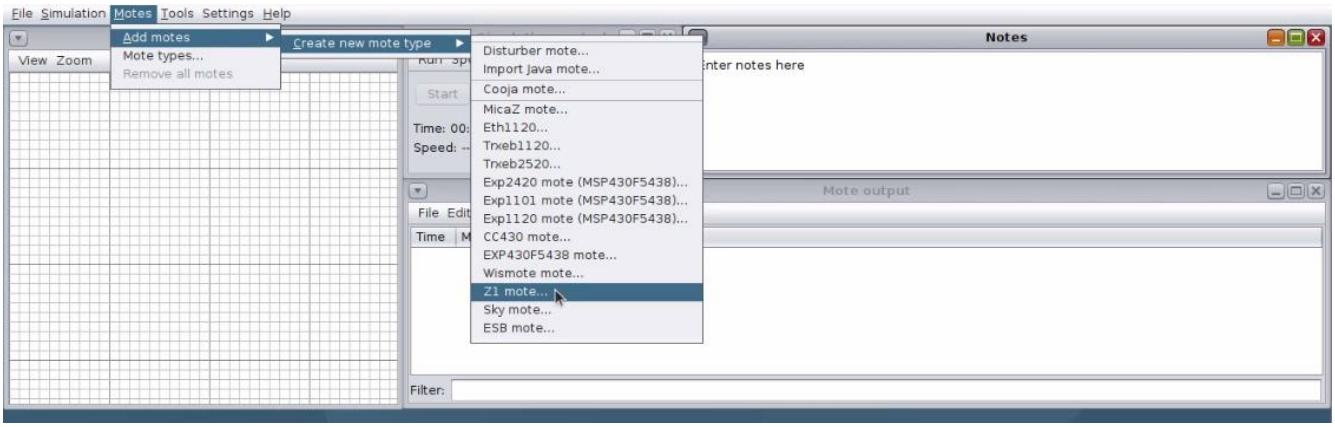


Figure34: Creation of Motes

After choosing the type of mote a window appears which allows you to give a description to the motes and to choose file which will be compiled as the compilation of a C program and this will allow you to simulate the mote

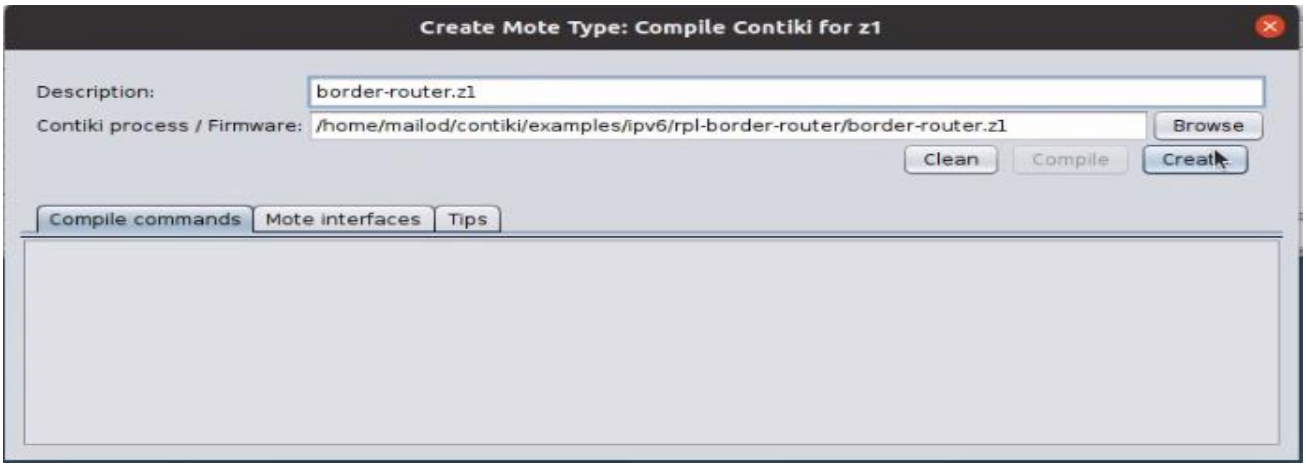


Figure 35:Compilation Window

we compiled (Compile) , once the compilation was finished, we create our mote (Create) and another window appears to give the number of motes that we want to create and their position appeared (we chose 10)

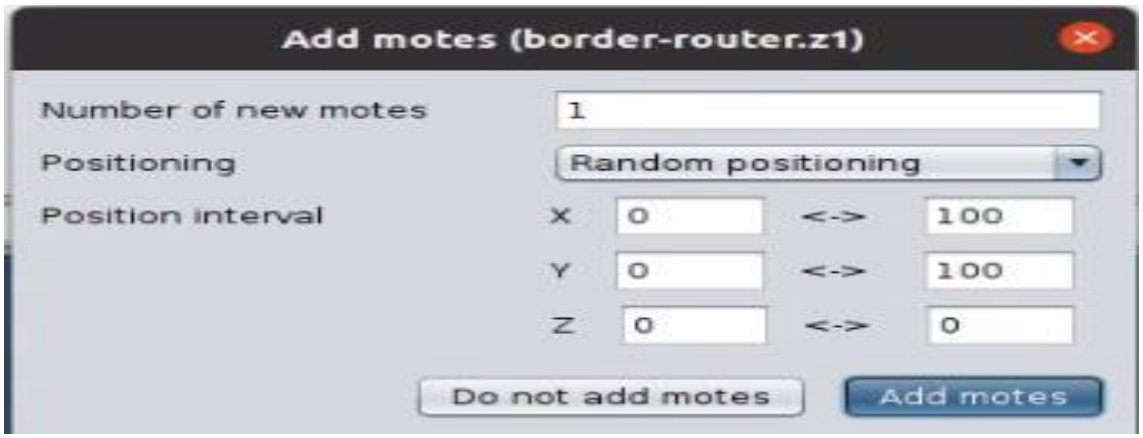


Figure 36: Adding Motes

A. The router: We gave it the description "broder router.z1" and selected the border-router.z1 file then

- **Information of border router :**

Figure 17: Border Router information

B. The server: same thing as the router except that we gave it the description "er-example-server.z1" and we selected the file er-example-server.z1

- **Information about server :**

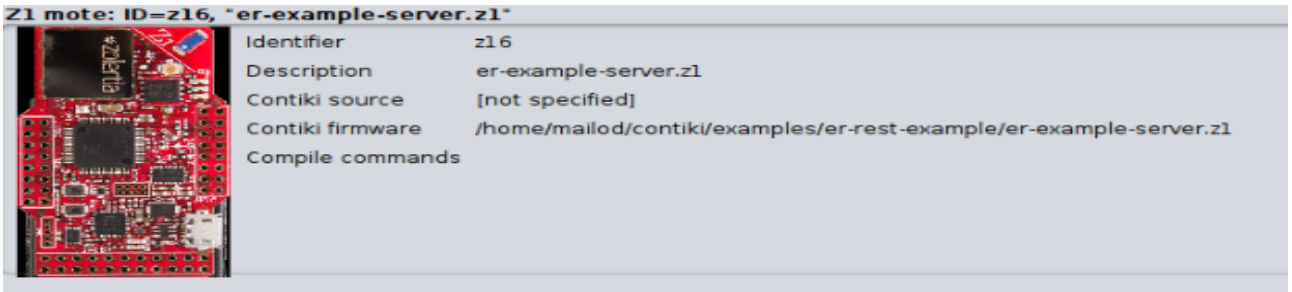


Figure 38: Server information

C. Clients We gave it the description “er-example-client.z1” and selected er-example-client.z1

- **Information about client :**

Figure 39: Client information

- This is the result of creating motes:

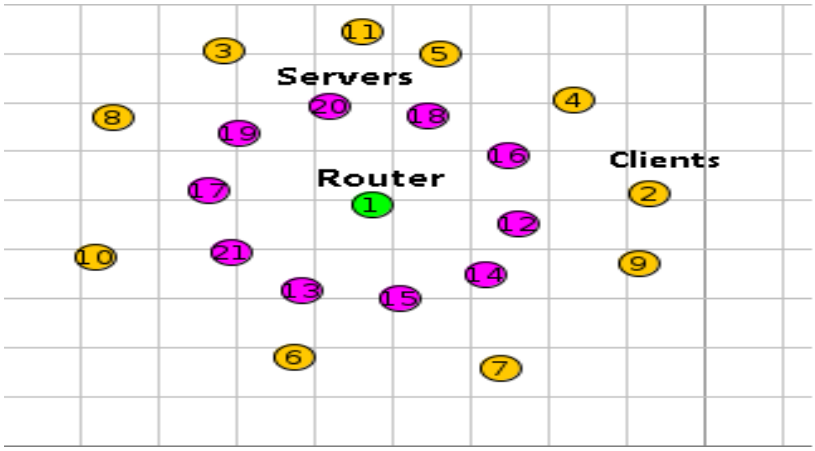


Figure 40: Network Display

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

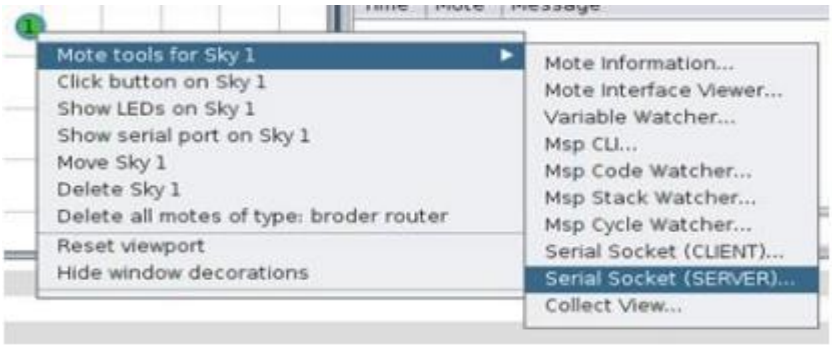


Figure 41: Access to the router window

- After a window with the start option is displayed .

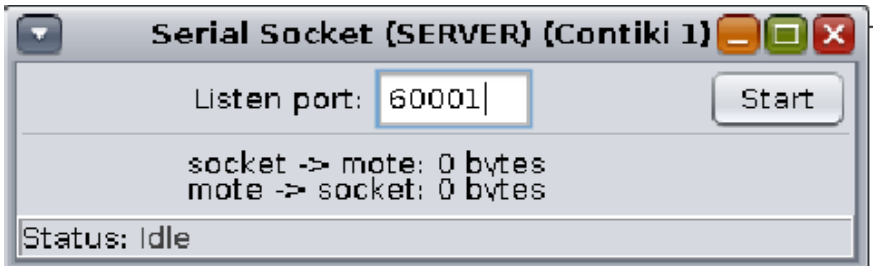


Figure 42: Serial Socket Server

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

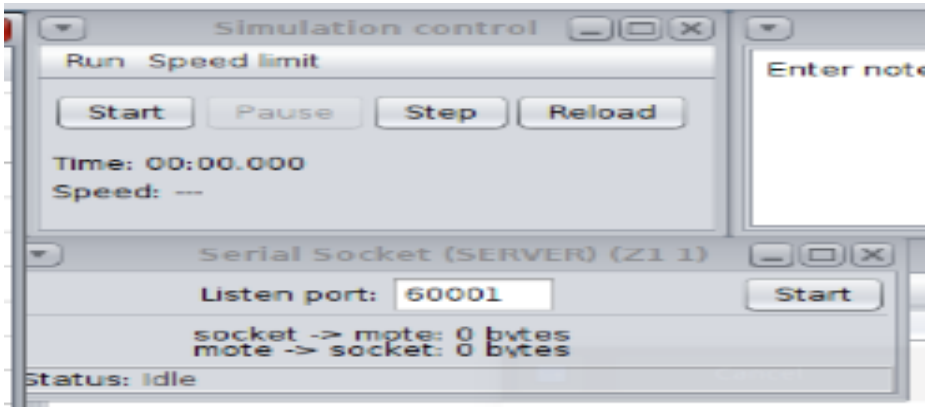
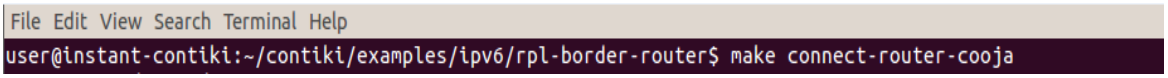


Figure 43: Starting The Simulation

- We also execute the command "**make connect-router-cooja**" to start



III.6.2 Scenario of the MQTT-sn protocol Simulation

For MQTT protocol we need 3 motes - Border router - Publisher - Subscriber

We will follow the same simulation steps as before(until we add a Border router node.

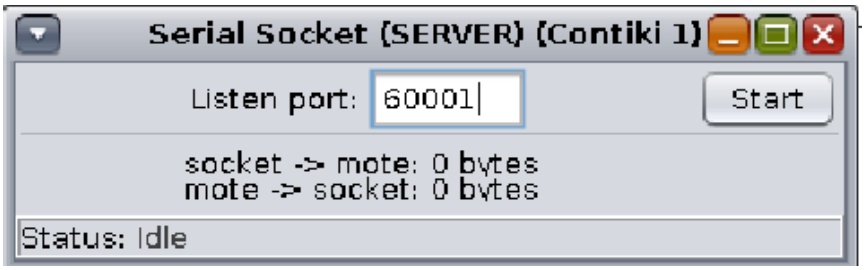
a) The Publisher : We gave it the description “Publisher” and selected main_core.z1**.**
b) Subscriber : We gave it the description “Subscriber” and selected main_core.z1.

Figure 44: Publisher information

Figure 18: Subscriber information

The two images represent the location of the two files in the Contiki -OS folder, and the red image on the left is for a small low-power development board used in Internet of Things projects and wireless sensor networks.

- Once the motes are created the router must be turned on



- We also execute the command "**make connect-router-cooja**" in terminal to start
- After that we open new terminal and run the command "**sudo ./broker_mqtts config.mqtt**"

```
mailod@mailod-Latitude-E6420:~/contiki/mqtt-sn-contiki/tools/mosquitto.rsmb/rsmb
/src$ sudo ./broker_mqtts config.mqtt
```

III.6.3 Performances evaluation experiments:

To evaluate the performance of COAP and MQT-SN protocols, this study used four parameters: the number of clients, the number of servers for CoAP , publisher and subscribe for MQTT-SN, data transmission interval and packet size. The impact of each parameter was examined in relation to the delay, power consumption, the number of packet delivered and throughput . It was done by simulating both protocols, including modifying the existing system files through custom code additions. For each change of a single parameter, the results were collected and plotted to facilitate protocol analysis.

Experiment 1 : Number of servers (CoAP) and Publishers (MQTT)

In this experiment we will measure the parameters we mentioned previously according to the number of servers and Publishers. The first is simulated by 10, then 20, 30, 40, 50, 60, 70, 100 servers and Publishers, Figures illustrate the simulation according to the number of the server. And packet size 10b and 10 Clients and subscribers and 5sec data transmission interval.

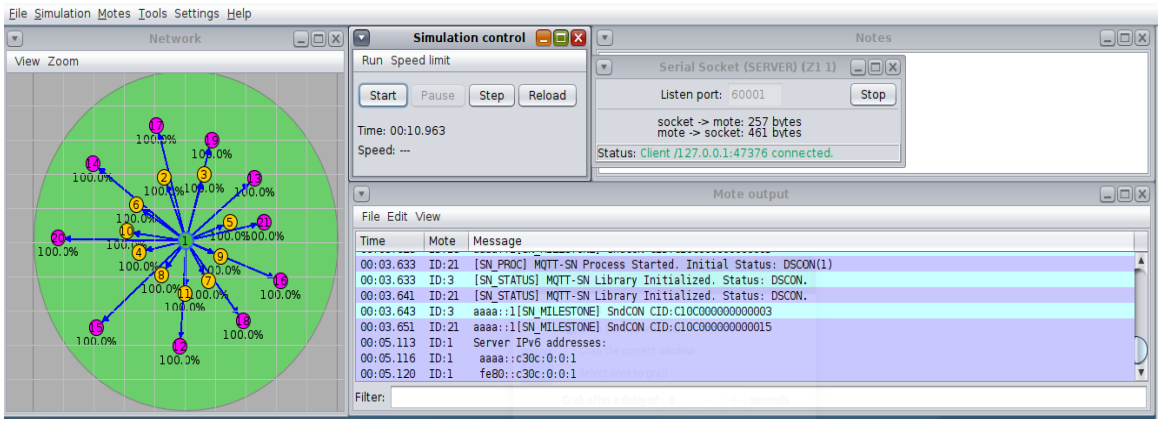


Figure 46: Simulation 10 server

Experiment 2 : Data Transmission Interval

In this experiment we will measure the parameters we mentioned previously according to transmission interval. The first is simulated by 5 seconds, then 10, 20, 30 seconds, with number 10 servers ,Publishers ,Clients , subscribers And 10b packet size.

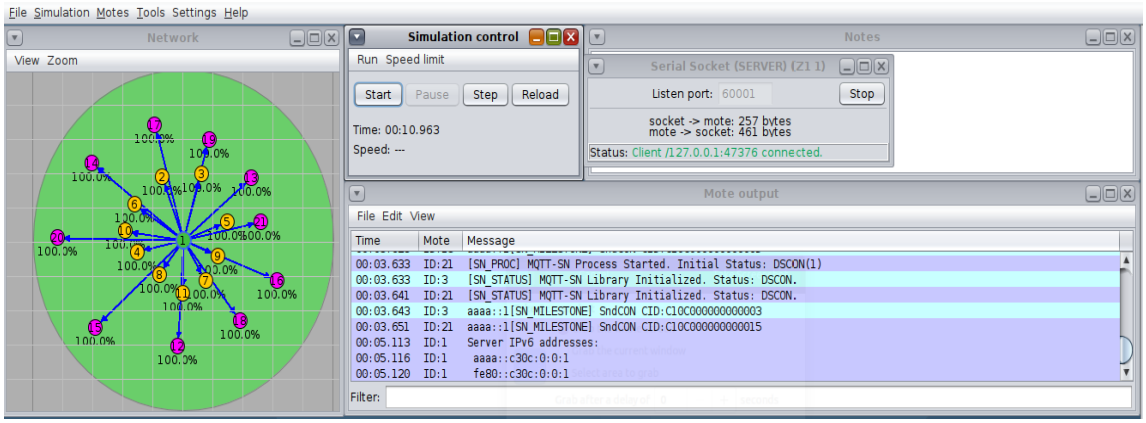


Figure 19: Simulation 5 Transmission Interval

Experiment 3 : Packet Size

In this experiment we will measure the parameters we mentioned previously according to Packet Size. The first is simulated by 10b, then 30, 50, 70, 90 byte, with number 10 servers ,Publishers ,Clients , subscribers and 5sec data transmission interval

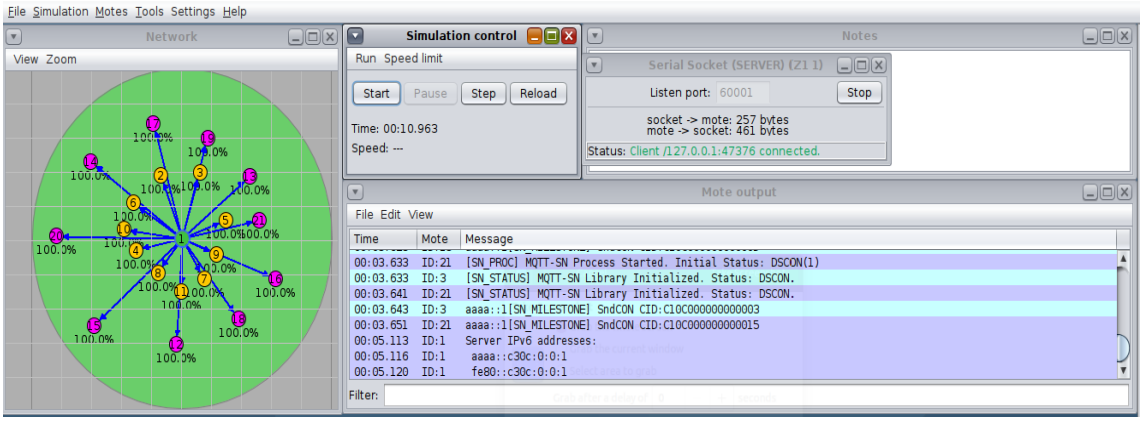


Figure 48: Simulation 10 byte packet size

Experiment 4 : Number of Clients (CoAP)and subscriber(MQTT)

In this experiment we will measure the parameters we mentioned previously according to the number of Clients and subscribers . The first is simulated by 10, then 20, 30, 40, 50, 60, 70, 100, Figures illustrate the simulation according to the number of the Clients. And packet size 10b and 10 servers and Publishers and 5sec data transmission interval.

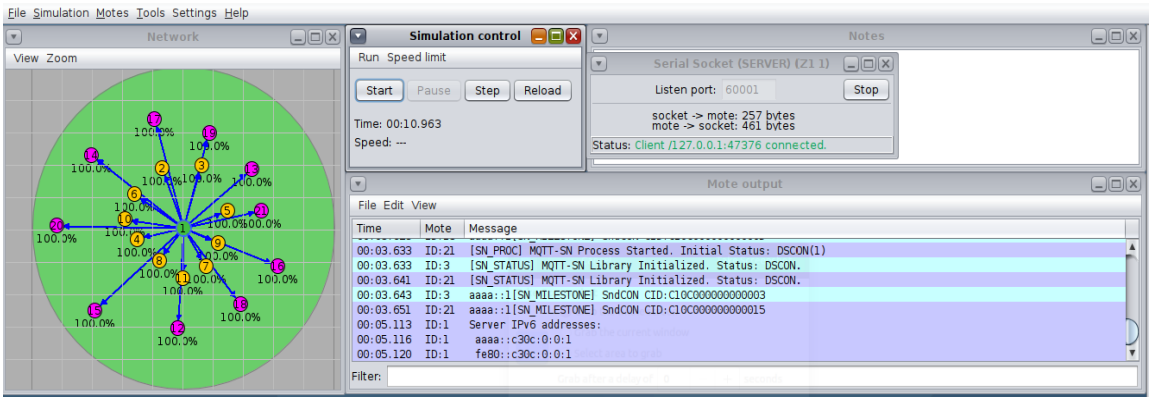


Figure 20: Simulation 10 Client and subscriber

III. 5 Conclusion

In this chapter, we have established and detailed the rigorous methodological framework that governed our simulation experiments. By carefully selecting the tools, designing the scenarios, and implementing them, we have succeeded in generating a large amount of primary data reflecting the performance of the two protocols. This data, in its raw form as log files, is the final product of this phase. However, no direct conclusions can be drawn from it. The next, and necessary, step is to process this data and transform it into a structured, analyzable form, which will be explained in the next chapter.

General Conclusion and Perspectives

The Internet of Things (IoT) is a transformative technological paradigm that is reshaping how we interact with our environment. As IoT devices proliferate—especially in resource-constrained settings—efficient, low-overhead communication protocols become essential for enabling scalable, reliable, and energy-conscious applications. This thesis focused on a systematic comparison of two leading IoT application-layer protocols: the Constrained Application Protocol (CoAP) and MQTT for Sensor Networks (MQTT-SN).

Through a structured, multi-phase methodology, this research provided both analytical and practical insights into protocol behavior. Simulations conducted in the Cooja environment using Contiki-NG OS allowed for the emulation of real-world constraints and performance conditions. Parameters such as payload size, transmission interval, node density, and network topology were varied, and their impact on critical performance metrics—latency, energy consumption, throughput, and packet delivery ratio—was measured. These results were compiled into structured datasets tailored for future predictive analysis and protocol selection using machine learning.

The methodology included the following phases:

- **Phase 1**: Literature review and protocol familiarization.
- **Phase 2**: Dataset design and generation for CoAP and MQTT-SN.
- **Phase 3**: Simulation of various network scenarios in COOJA.
- **Phase 4**: Data visualization and metric analysis using Python.
- **Phase 5**: Comparative interpretation of results across performance indicators.

The findings revealed that MQTT-SN generally outperformed CoAP in terms of energy efficiency and scalability, particularly in dense networks. CoAP, on the other hand, was more suitable for lightweight, low-latency scenarios and supported native resource discovery features. These observations can guide IoT designers in selecting the most appropriate protocol for specific application demands.

Perspectives for Future Work

While the presented framework offers a reproducible and detailed analysis, several opportunities exist for extending this research:

1. **Physical Testbed Validation**: Future work can deploy CoAP and MQTT-SN on real IoT hardware platforms to validate simulation results and account for unpredictable environmental variables such as interference and mobility.
2. **Complex Network Conditions**: Extending simulations to incorporate high-congestion scenarios, dynamic topologies, and heterogeneous devices could provide a more comprehensive evaluation.
3. **Security and QoS Metrics**: Integrating evaluations of protocol-level security and Quality of Service (QoS) under stress conditions would broaden the applicability of the findings.
4. **Machine Learning Integration**: The generated datasets are well-suited for training ML models that predict protocol behavior or suggest optimal configurations based on given environmental and application parameters.
5. **Adaptive Protocol Middleware**: Future systems could use predictive models to implement adaptive protocol switching, improving performance dynamically in real-time applications.

In summary, this thesis contributes a robust simulation-based evaluation of two leading IoT protocols and introduces a novel, structured dataset that supports future research in machine learning and intelligent network optimization. These tools and insights are a step forward in the development of efficient, scalable, and context-aware IoT communication systems.

Bibliography

[01] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," Comput. Netw., vol. 52, no. 12, pp. 2292–2330, Aug. 2008.

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

[03] R. Tafazolli, Ed., Technologies for the Wireless Future. Hoboken, NJ, USA: Wiley, 2006.

[04] O. Taleb and A. Mankouri, "Programmation de la sécurité Internet des Objet, Etude de cas module WIFI Electric imp," M.S. thesis, Univ. of Tlemcen, Tlemcen, Algeria, 2016.

[05] S. Tyagi, "How to Design a Scalable ChatGPT-like AI System," Medium, Jan. 3, 2023. [Online]. Available: https://sonu-tyagi.medium.com/how-to-design-a-scalable-chatgpt-like-ai-system-e724f1bdc4c7. (Accessed: Apr. 15, 2025).

[06] M. Y. Atoumi and S. Bensadi, "Approche évolutionnaire pour la composition de services sensible à la QoS dans l'Internet des Objets à large échelle," M.S. thesis, Univ. of Bejaia, Bejaia, Algeria, 2018.

[07] Y. Challal, "Sécurité de l'Internet des Objets : vers une approche cognitive et systémique," Ph.D. dissertation, Univ. de Technologie de Compiègne, Compiègne, France, 2012.

[08] Digora, "Définition IoT et stratégie IoT : tout savoir sur l'Internet of Things," Digora Blog. [Online]. Available: https://www.digora.com/fr/blog/definition-iot-et-strategie-iot. (Accessed: Apr. 22, 2025).

[09] Connectwave, "Comment se compose un système IoT?," Connectwave, 2022. [Online]. Available: https://www.connectwave.fr/techno-appli-iot/iot/reseaux-et-infrastructuresiot/. (Accessed: May 5, 2025).

[10] J. Stankovic, "Wireless Sensor Networks," Computer, vol. 41, no. 10, pp. 92–95, Oct. 2008.

[11] S. Rabeb, "Modèle collaboratif pour l'Internet of Things (IoT)," M.S. thesis, Univ. du Québec à Chicoutimi, Chicoutimi, QC, Canada, 2016.

[12] I. Saleh, "Internet des Objets (IdO) : Concepts, Enjeux, Défis et Perspectives," Internet des objets, vol. 2, no. 1, 2018.

[13] Wikiwai, "Internet des Objets : architectures, protocoles et applications," Wikiwai, Feb. 21, 2020. [Online]. Available: http://www.wikiwai.com/2020/02/21/internet-des-objets-architectures-protocoleset-applications/. (Accessed: May 11, 2025).

[14] "A Comprehensive Study on Intrusion Detection and Prevention Systems in IoT Networks," in Unknown Book Title, Springer, n.d., ch. 1. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-97-5624-7_1. (Accessed: May 18, 2025).

[15] S. Cirani, G. Ferrari, M. Picone, and L. Veltri, Internet of Things: Architectures, Protocols, and Standards. Hoboken, NJ, USA: Wiley, 2019.

[16] P. Pedamkar, "Introduction to IoT | A Comprehensive Guide to Internet of Things (IoT)," EDUCBA. [Online]. Available: https://www.educba.com/introduction-to-iot/. (Accessed: May 2, 2025).

[17] Webchoice, "Fundamental characteristics that make the Internet Of Things what it is," Webchoice. [Online]. Available: https://www.webchoiceonline.com.au/fundamental-characteristics-that-makesthe-internet-of-things-what-it-is/. (Accessed: May 20, 2025).

[18] Emnify, "A Comprehensive Guide to IoT Protocols," Emnify. [Online]. Available: https://www.emnify.com/iot-glossary/guide-iot-protocols. (Accessed: Apr. 29, 2025).

[19] S. Feng et al., "Sécurité des objets Connectés," Institut national des hautes études de la sécurité et de la justice, Paris, France, Rep., 2014. [Online]. Available: https://www.cigref.fr/archives/entreprises-et-culturesnumeriques/wp/wp-content/uploads/2014/12/rapport-auditeurs-cigref-inhesj-securiteobjets-connectes.pdf.

[20] MQTT.org, "MQTT - The Standard for IoT Messaging." [Online]. Available: https://mqtt.org/. (Accessed: Apr. 25, 2025).

[21] OASIS, "MQTT Version 5.0," OASIS Standard, Mar. 2019. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html. (Accessed: Apr. 30, 2025).

[22] Y. Melka, "Model-Based Testing des applications du protocole MQTT," SlideShare. [Online]. Available: https://www.slideshare.net/ymelka/model-based-testing-des-applications-du-protocole-mqtt. (Accessed: May 7, 2025).

[23] Z. Shelby, K. Hartke, and C. Bormann, “The Constrained Application Protocol (CoAP),” RFC 7252, Internet Engineering Task Force, Jun. 2014. [Online]. Available: https://www.rfc-editor.org/info/rfc7252.

[24] A. Banks, E. comm. OASIS, and R. Gupta, “MQTT Version 5.0,” OASIS Standard, Mar. 2019. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html.

[25] W. McKinney, Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, 2nd ed. O’Reilly Media, 2017.

[26] C. Sengul and A. Kirby, "Message Queuing Telemetry Transport (MQTT) and Transport Layer Security (TLS) Profile of Authentication and Authorization for Constrained Environments (ACE) Framework," RFC 9431, Jul. 2023. [Online]. Available: https://www.rfc-editor.org/info/rfc9431.

[27] W. Korichi, "Partage De Données En Environnements Mobiles Ad Hoc," Magister thesis, Dept. Comput. Sci., Kasdi Merbah Univ., Ouargla, Algeria, 2013.

[28] Univ. Paris-Sud, "Introduction A La Programmation En Langage Python," Licence MPI S2 course material, 2016.

[29] L. B. Saad, C. Chauvet, and B. Tourneau, "Simulation of The Rpl Routing Protocol for Ipv6 Sensor Networks: Two Cases Studies," in Proc. 2011 Int. Conf. Sensor Technol. Appl. (SENSORCOMM), French Riviera, France, Aug. 2011, pp. 205–211.