



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

Memory

With a goal to obtaining the diploma of

Master degree

Presented by :

Soudani M. Taha

TITLE:

**PERFORMANCE EVALUATION OF
MOBILITY IN IoT PROTOCOL**

Before the jury:

Dr.Azzaoui.N
Dr. Cheradid.A
KHELILI Khalida.f

MCA President
MAA
MAA Supervisor

UKM Ouargla
UKM Ouargla
UKM Ouargla

Thanks

I am instructed to thank your grace that bestowed upon me and my father and to work

You will be pleased with him, and enter me with your mercy on your righteous servants. "

Praise be to God, so much praise for his innumerable blessings, and from him he has succeeded us to complete

This work, where it was bounty to us, was great.

In gratitude we extend our sincere thanks, sincere praise and sincere appreciation to all of us

Help us accomplish this work and especially mention Professor Khalili.KF for her acceptance

Supervising this modest study with all effort and time with care, guidance and support.

We also do not forget the professors of the Department of Computer Science for their advice and guidance.

Finally, we are pleased to extend our sincere thanks and appreciation to all those who helped us little or much, hoping for them

From God Almighty reward, reward and praise, and praise be to God, Lord of the worlds.

Dedication.

Praise be to Allah. Alhamdulillah for everything after a study process that has brought with it many difficulties, hardship and fatigue. I dedicate my graduation to my hope in life and the joy of my eyes and the secret of my success, my dear mother, may God last and prolong her life. And my father, who taught me how to hold the pen and how to write words without regret. To everyone in my heart, just as I do not forget all those who supported us in this suffering from the Corona pandemic to the scientific and technical edifice and the medical staff, especially to the Qasdimerbah University. I dedicate this work to all of them.

Abstract:

The Internet of things is defined as connecting various things to the Internet, such as a phone, refrigerator, traffic lights, smart home, street lights, and even the person himself, so that these things can communicate with each other and complete various tasks without the need for human intervention. The purpose of this technology is to make people's lives safer and more luxurious, while also saving them time and effort. IoT technologies are being used in a variety of fields, including health care, agriculture, manufacturing, self-driving cars, smart homes and cities, in other side we can talk about the movement of IoT devices which is considered as a mobility devices that has on IoT fields. And because Internet of Things devices are small and limited in terms of energy, processing power, and battery life, and rely on wireless sensor networks (WSN) that are characterized by low-bandwidth and unreliable communication, many of these devices cannot provide efficient and acceptable communication. To facilitate IoT connectivity, numerous protocols were proposed on the application layer (MQTT, AQMP, CoAP..., with CoAP being the most widely used), and because of the variety of protocols (no standardization), selecting a protocol for an application is difficult. This work will provide a performance study of one of the most commonly used protocols: CoAP, to assist users in their choice of communication protocol. In this work we have evaluated the mobility in CoAP protocol comparing it with origin CoAP protocol, by evaluating the parameters: energy consumption, packet loss and delays, by simulating the protocol using the COOJA simulator, and studying the results obtained from the simulation then analyzing them to find its weakness and advantages in both sides with and without mobility.

In our study we focused on mobility in CoAP, by try different topologies and different numbers of motes, and by analyzing results between both experiments we will judge the CoAP protocol after that.

Keywords: Internet of Things, CoAP, Performances Evaluation, Simulation, COOJA, WSN.

Résumé:

L'internet des objets se définit comme la connexion de divers objets à l'internet, tels qu'un téléphone, un réfrigérateur, des feux de circulation, une maison intelligente, des lampadaires et même la personne elle-même, afin que ces objets puissent communiquer entre eux et accomplir diverses tâches sans nécessiter d'intervention humaine. L'objectif de cette technologie est de rendre la vie des gens plus sûre et plus luxueuse, tout en leur faisant gagner du temps et des efforts. Les technologies de l'IdO sont utilisées dans divers secteurs, notamment les soins de santé, l'agriculture, la fabrication, les voitures à conduite autonome, les maisons et les villes intelligentes, etc. Et comme les appareils de l'Internet des objets sont petits et limités en termes d'énergie, de puissance de traitement et d'autonomie, et qu'ils s'appuient sur des réseaux de capteurs sans fil (WSN) caractérisés par une faible bande passante et une communication peu fiable, beaucoup de ces appareils ne peuvent pas assurer une communication efficace et acceptable. Pour faciliter la connectivité de l'IdO, de nombreux protocoles ont été proposés sur la couche application (MQTT, AQMP, CoAP...., CoAP étant le plus utilisé), et en raison de la variété des protocoles (pas de normalisation), la sélection d'un protocole pour une application est difficile. Ce travail fournira une étude de performance de l'un des protocoles les plus couramment utilisés : CoAP, afin d'aider les utilisateurs dans leur choix de protocole de communication. Dans ce travail, nous avons évalué la mobilité dans le protocole CoAP, en évaluant les paramètres : consommation d'énergie, débit, perte de paquets et délais, en simulant le protocole en utilisant le simulateur COOJA, et en étudiant les résultats obtenus à partir de la simulation puis en les analysant pour trouver ses faiblesses et ses avantages. Nous avons conclu, à la fin de notre travail, que le protocole CoAP possède des qualités d'interopérabilité, de fiabilité et d'évolutivité. Une de ses faiblesses est la consommation d'énergie en abondance.

Mots-clés : Internet des objets, CoAP, Évaluation des performances, Simulation, COOJA, WSN.

الملخص :

يُعرّف إنترنت الأشياء بأنه ربط أشياء مختلفة بالإنترنت ، مثل الهاتف ، والثلاجة ، وإشارات المرور ، والمنزل الذكي ، وأضواء الشوارع ، وحتى الشخص نفسه ، بحيث يمكن لهذه الأشياء التواصل مع بعضها البعض وإكمال المهام المختلفة بدون الحاجة إلى التدخل البشري. الغرض من هذه التقنية هو جعل حياة الناس أكثر أمانًا، مع توفير الوقت والجهد أيضًا. تُستخدم تقنيات إنترنت الأشياء في مجموعة متنوعة من الصناعات ، بما في ذلك الرعاية الصحية والزراعة والتصنيع والسيارات ذاتية القيادة والمنازل والمدن الذكية وما إلى ذلك. ونظرًا لأن أجهزة إنترنت الأشياء صغيرة ومحدودة من حيث الطاقة وطاقة المعالجة وعمر البطارية ، وتعتمد على شبكات الاستشعار اللاسلكية (WSN) التي تتميز بنطاق ترددي منخفض واتصالات غير موثوقة ، فإن العديد من هذه الأجهزة لا يمكنها توفير كفاءة وفعالية. اتصال مقبول لتسهيل اتصال إنترنت الأشياء ، تم اقتراح العديد من البروتوكولات على طبقة التطبيق (MQTT ، AQMP ، CoAP ، مع كون CoAP هو الأكثر استخدامًا) ، وبسبب تنوع البروتوكولات (بدون توحيد) ، اختياري بروتوكول لمجموعة التطبيق صعب. سيوفر هذا العمل دراسة أداء لواحد من أكثر البروتوكولات استخدامًا: CoAP ، لمساعدة المستخدمين في اختيار بروتوكول الاتصال. في هذا العمل ، قمنا بتقييم التنقل في بروتوكول CoAP ، من خلال تقييم المعلمات: استهلاك الطاقة ، والإنتاجية ، وفقدان الحزمة والتأخير ، عن طريق محاكاة البروتوكول باستخدام محاكي COOJA ، ودراسة النتائج التي تم الحصول عليها من المحاكاة ثم تحليلها للعثور عليها. الضعف والمزايا لقد استنتجنا ، في نهاية عملنا ، أن بروتوكول CoAP يحقق قابلية التشغيل البيئي والموثوقية وقابلية التوسع. واحدة من نقاط ضعفها هو استهلاك الطاقة بكثرة.

الكلمات المفتاحية: إنترنت الأشياء ، CoAP ، الأداء ، التقييم ، المحاكاة ، COOJA ، WSN.

Table of contents

THANKS.....	II
Dedication1	III
Abstract:	IV
Table of contents	V
Liste of tables	VI
Liste Of figures.....	VII
List of Abreviations.....	VIII
INTRODUCTION.....	5

CHAPETR 1: INTERNET OF THINGS

1	INTRODUCTION.....	9
2	THE INTERNET OF THINGS DEFINITION	9
3	DOMAINES OF THE IoT	9
4	INTERNET OF THINGS TOOLS.....	12
4.1	The hardware side of IoT	12
4.2	The software side of IoT.....	13
5	THE INTERNET OF THINGS ARCHITECTURE	13
6	Massaging Protocols for IOT System	15
6.1	COAP (<i>Constrained Application Protocol</i>)	15
6.2	MQTT (<i>Message Queue Telemetry Transport</i>)	15
6.3	HTTP (<i>Hyper Text Transport Protocol</i>)	16
6.4	AMQP (<i>Advanced Message Queuing Protocol</i>)	17
7	Comparative Analysis of IOT Protocols.....	18
8	Standards And Standardization.....	19
9	CONCLUSION	

CHAPETR 2: CoAP PROTOCOL Theoretically

1	INTRODUCTION.....	23
2	DEFFINITION OF CoAP (CONSTRAINED APPLICATION PROTOCOL).....	23
3	FEATURES CoAP HAS THE FOLLOWING MAIN FEATURES	23
4	CoAP ARCHITECTURE.....	24

4.1	Message Layer.....	25
4.2	Request/Response Layer	26
5	MESSAGES FORMAT	29
6	PROTOCOL CHARACTERISTICSAN.....	32
6.1	Caching.....	32
6.2	Proxying.....	33
6.3	Resource Discovery	35
7	EXAMPLE OF COMMUNICATIONS IN CoAP	36
8	FUNCTIONAL APPLICATIONS OF CoAP PROTOCOL.....	37
9	SECURING CoAP	38
10	CoAP AND IOTASPACT.....	38
10.1	Energy Consumption	39
10.2	Infrastructure communication	39
10.3	Reliability	41
10.4	Scalability.....	41
10.5	Compatibility	41
10.6	Performances	42
10.7	Effectiveness charge.....	42
11	CONCLUSION	43

CHAPETR 3: CoAP PROTOCOL SIMULATION

1	INTRODUCTION.....	45
2	VALIDATION METHODS.....	45
2.1	Analytical methods	45
2.2	Real Experience.....	45
2.3	Simulation	46
3	METHODOLOGY	46
4	TOOLS USED IN THIS SIMULATION.....	46
4.1	Software	46
4.2	Hardware	47
5	SIMULATION ENVIRONMENT.....	49
6	SIMULATION SCENARIO: SMART PRISON MOVMENT	50
7	CoAP SIMULATION USING COOJA	51
8	MOBILITY OF NODE IN COOJA	55
8.1	Downloading and Displacing Mobility.....	55
8.2	Modifications and Changes on Mobility.....	55
8.3	Enabling the Mobility Plugin in Cooja.....	57
9	PERFORMANCES EVALUATION EXPERIMENTS.....	59
9.1	Experiment 1: Fixed nodes	59
9.2	Experiment 2: Mobility of nodes	59
10	CONCLUSION	62

CHAPETR 4: PERFORMANCES EVALUATION

1	INTRODUCTION	64
2	CRETERIAS OF EVALUATION	64
2.1	Energy Consumption	64
2.2	Packet loss	65
3	PERFORMANCE EVALUATION METHODE	65
3.1	Results Simulation Experiment1: Fixed nodes	66
3.2	Results Simulation Experiment 2: Mobility nodes.....	68
4	EVALUATION RESULTS.....	79
5	CONCLUSION	81
	CONCLUSION... ..	83
	Bibliography Documents	85
	Web Bibliography.....	89

List of tables

Table 01: Comparative Analysis of Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP.....19

Table 02 : Organizations Standards de protocole IoT.....20

Table 03: CoAP respond codes.....30

Table 04: Functional Applications of CoAP protocol.....39

Table 05: simulation environment.....51

Table 06: Characteristics of the Tmote Sky sensor node platform.....52

Table 07: Critecia values depended on Clients and Servers number.....68

Table 08: Critecia values depended on mobility of Clients and Servers number.....70

Table 09: compare standards by ratio for different transmissions72

Table 10: CoAP and IoT Aspect Results.....73

List Of figures

Figure 01: Efficient waste management in Smart Cities.....	11
Figure 02. Efficient and effective collaborative research supported by sensing as a service model.....	12
Figure 03. Smart Water Management.....	13
Figure04: Deferent IoT Architecture.	15
Figure 05: CoAP protocol.....	16
Figure 06: MQTT protocol	17
Figure 07: HTTP protocol.....	18
Figure08: AMQP protocol.....	19
Figure 09: CoAP architecture.....	25
Figure 10: Reliable message transmission (RST)	26
Figure 11: Reliable message transmission (RST)	26
Figure 12: Piggy-backed request/response transmission.....	27
Figure 13: Separate request/response transmission	28
Figure 14: Reliable message transmission	28
Figure 15: CoAP message format.....	30
Figure 16: methods and codes.....	31
Figure 17: Option Format.....	31
Figure 18: Caching in CoAP.....	34
Figure 19: Network Architecture Using Proxy.....	34
Figure 20: Protocol Stack.....	35
Figure 21: Example Constrained RESTful Environmnt (CoRE) direct resource discovery &(CoAP)request.....	36
Figure 22: An example resource discovery by using a Resource Directory.	37
Figure 23: CoAP example communication.....	38
Figure 24: Abstract Layering of DTLS-Secured CoAP.....	40
Figure 25: COOJA simulator Interface.....	49
Figure26: COOJA window.....	53

Figure27: Border Router Mote.....	54
Figure28: CoAP server mote.....	54
Figure29: CoAP client mote.....	55
Figure30: border router and CoAP server.....	55
Figure31: Libcoap file.....	56
Figure32: Communication CoAP Server.....	56
Figure33: displacement of mobility file.....	57
Figure34: Change extension in build.xml file.....	57
Figure35: changing cooja path in cooja.config.....	58
Figure36: changing cooja path in Mobility.java file.....	58
Figure37: activating mobility file in cooja.....	58
Figure38: Adding mobility to cooja extensions.....	59
Figure39: Adding mobility path to cooja simulator.....	59
Figure40: Enabling mobility tool.....	60
Figure41: mobility tool.....	60
Figure42: Simulation For 20, 50 clients (linear, square).....	61
Figure43: Add Modifications To er-coap-engine.c file.....	62
Figure44: Add Modifications To powertrace.c file.....	63
Figure45: Add Modifications To er-example-server and er-client-example.....	63
Figure46: The final results are in Mote output.....	67
Figure 47: Curves of Packet loss against the Number of COAP Clients and Topology.....	68
Figure 48: energy consumption values in relation to the number of Clients and Topologies.....	69
Figure 49: curves of packet loss against the number and mobility of motes.....	70
Figure 50: energy consumption values in relation with mobility nodes.....	71

List of Abbreviations:

IEEE	Institute of Electrical and Electronics Engineers
IETF	International Engineering Task Force
CoAP	Constrained Application Protocol
HTTP	Hypertext Transfer Protocol
IoT	Internet Of Thing
M2M	Machine To Machine
MQTT	MQ Telemetry Transport
RFID	Radio Fréquence Identification
IP	Internet Protocole
IPV6	Internet Protocole version 6
IPV4	Internet Protocole version 4
6LoWPAN	over Low Power Wireless Personal Area Networks
MAC	Medium Access Control
WSN	Wireless Sensor Networks
AMQP	Advanced Message Queuing Protocol
TCP/IP	Transmission Control Protocol / Internet Protocol
UDP	User Datagram Protocol
RAM	Random Access Memory
WPAN	Wireless Personal Area Network
CPU	central processing unit
Wifi	wireless fidelity
HTML	hyper text markup language
LLN	Low-power and Lossy Network
RPL	Routing Protocol for Low-power and Lossy Network
URL	Uniform Resource Locator
TLS	Transport Layer Security
WOT	web of thing
DTLS	Datagram Transport Layer Security
ROM	Read-Only Memory.
DR	resource discovery



General Introduction

General Introduction:

From the computer to the Internet to the laptop to the smartphone, the development of informatics and technology has gone through many stages that have changed the world for the better, where our connection is no longer limited to the click of a button and is now possible for a variety of purposes, like (refrigerators, TVs, traffic light ... etc.). These objects can communicate with one another to accomplish certain jobs and functions without the need for human intervention, and we now refer to this as the Internet of Things (IoT). In most industries, including health, industry, agriculture, smart homes, smart cities, self-driving cars, and others, the utilization of IoT technology has proven to be dependable, as we could mention the mobility in our study that it is considered as a useful side out of IoT domain. The Internet of Things apps must adhere to all of the technology's properties, including reliability, interoperability, scalability, and security, in order to use it. However, due to the nature of connected devices and the emergence of complex networks in the Internet of Things, maintaining these features is difficult. We may classify every device linked to the Internet as a smart thing, or anything that can get an IPv6 or IPv4 address. The sensing devices, such as motion sensors, humidity sensors, and gas sensors, are among the most critical components of a smart object. According to the task that the smart object must perform and the nature of these devices, (constrained and limited Resources in terms of energy, processing power, memory capacity, and battery) it works on the wireless sensor networks (WSN), Low-power and unreliable communication, as well as intermittent connections, define these systems. From here, one of the most significant difficulties in the Internet of Things emerges: how to support high-efficiency and reliable machine-to-machine communication. Several protocols (CoAP, MQTT, AMQP ...etc.) have been proposed in order to obtain the best connection, but their implementation must adhere to the Internet of Things' standards and meet all of the required qualities. There hasn't been a fundamental protocol that incorporates all standards and characteristics until now. The variety and difficulty of selecting the optimal communication protocol for IoT applications appears here for users. In recent years, it is widely expected that CoAP will become the standard protocol in the Internet of Things.

In this study, we will use simulation and a COOJA network simulator to assess the performance of the CoAP protocol. allows us to run simulations with low bandwidth and high latency, as well as network loss packets, to ensure that the system is functioning efficiently and meeting the essential specifications (characteristics internet of things). Through the results of the experiments, we can gain insight into the protocol's effectiveness within a limited wireless network using metrics acquired from the simulation.

This paper research will be organized as follows:

In Chapter 1, We will demonstrate various Internet of Things principles. Introduces its definition, as well as some of the industries that use IoT applications, architecture, and the most often used protocols. Finally, we discuss some of the Internet of Things problems (aspects).

In Chapter 2, the CoAP protocol will be investigated theoretically. We will look at the definition, features, structure, and characteristics of the protocol, as well as the applications that use it, and we examine the opposite of the Internet of Things characteristics.

Chapter 3: In this part, we will examine the effectiveness of mobility on the CoAP protocol in a COOJA simulator through some experiments based on smart prison management, using various criteria and standards.

Chapter 4: This section contains the simulation results, which will be used to evaluate the performance of the CoAP protocol either with or without mobility in connection to Internet of Things features.



CHAPTER 01

INTERNET OF THINGS

Chapter 01: INTERNET OF THING

01.Introduction:

The phrase "Internet of Things" is defined by a variety of professional and scientific research sources. Kevin Ashton, co-owner and CEO of Auto-ID Center, coined the term "internet of things" in 1999. Various professional standards agencies, organizations and associations in the field of IK technologies, as well as numerous scholars, have defined the notion of IoT as it has evolved and expanded in application.

The Internet of Things (IoT) concept can be thought of as a way to augment existing human-application interaction by adding a new level of integration and communication represented by objects. The potential of the Internet of Things concept allows for its adoption and application in a variety of domains in application, including society, the environment, and industry.

02.Definition:

The Internet of Things (IoT) radically alters how technology is integrated into our physical environments. To this purpose, the Internet of Things promotes a vision in which everything produces/consumes, processes, and exchanges of data. Through variety of application fields, individuals engage with and benefit from such immersive and pervasive technologies. In a wide range of fields, IoT solutions have yet to be commercialized.[2]

Therefore, the number of data transmissions in IoT systems is smaller than it could be comparing with the potential.As a result, systems send messages at a low-frequency. The use of a reduced frequency of message transfer between endpoint devices is associated with a lower risk of data traffic.[3]

03.Domaines of IOT:

3.1 Waste management:

One of the most difficult challenges that modern cities face is waste management. Collection, transport, processing, disposal, management, and monitoring of waste materials are all part of waste management. These procedures are expensive in terms of money, time, and labor. Streamlining it operations allows to save money that may be used toward other pressing issues. In the waste management area, we demonstrate how the sensing as a service approach works. Instead of building sensors and collecting data on their own, the sensing as a service model allows all interested parties to share the infrastructure and share the expenses. The most crucial component of such a partnership is the cost savings that would otherwise be necessary for separate parties. To achieve their own goal, all

interested parties can obtain and process sensor data in real time. The price is determined by the interest group's data requirements.[4]

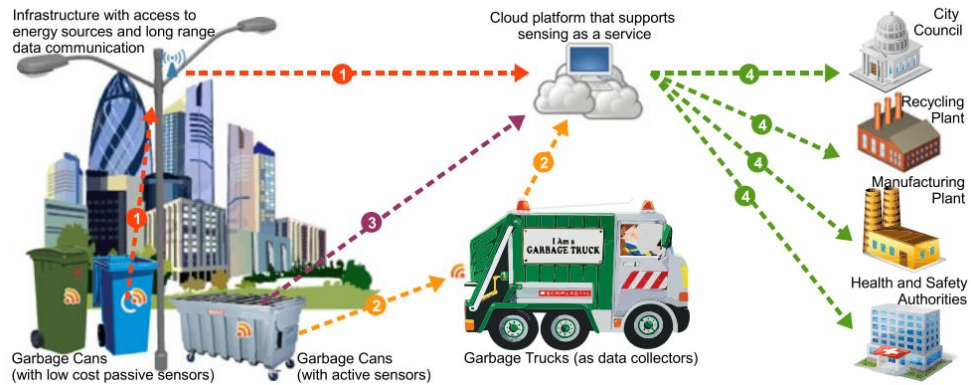


Figure 01: Efficient waste management in Smart Cities

3.2 Agricultural system:

The searchers are working on two projects: Phenonet [5] and OpenIoT [6], in which they are building and implementing open platforms for sensor data collection, processing, and sharing in the agricultural sector. In this situation the general public is not directly involved. And inside the smart agricultural area, they demonstrate how the sensing as a service paradigm works. It is an essential component of smart city because it contributes the food supply chain that enables a large number of communities clustered in cities to grow. Examine the Phenonet project to see how the sensing as a service concept could be applied. A sensor network called Phenonet collects data from an experimental area. The High-Resolution Plant Phenomics Centre is putting a network of smart sensor nodes to a test, which can monitor plant development as well as weather conditions. Despite the fact that the major goal of using sensors and collecting data is to better understand plant development in different climates.

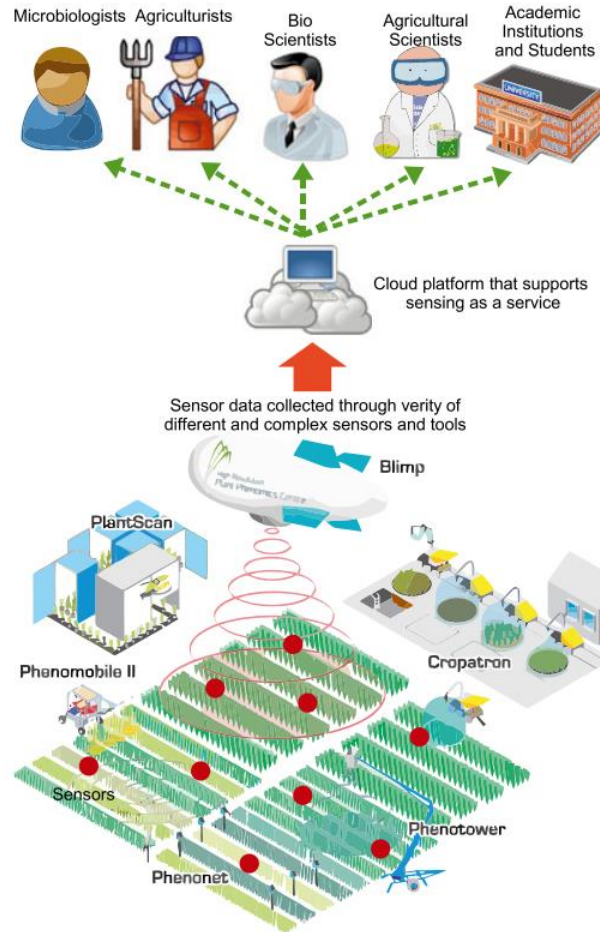


Figure 02. Efficient and effective collaborative research supported by sensing as a service model.

3.3 Smart water management:

One of the most significant parts of future smart cities is the city's water infrastructure. In smart water management at city sizes, effective and energy-efficient transportation and use of water, along with cost-effective and environmentally friendly wastewater treatment, are crucial.

A smart water system is supposed to assist the monitoring of city-wide water usage, transportation, and future water use forecasting, among other things. specifies the requirements for water harvesting and groundwater monitoring, which will rely on Fog/Cloud computing infrastructure such as wireless sensors, smart meters, GPS devices, Fog gateways, Cloud platforms, and so on.[7]

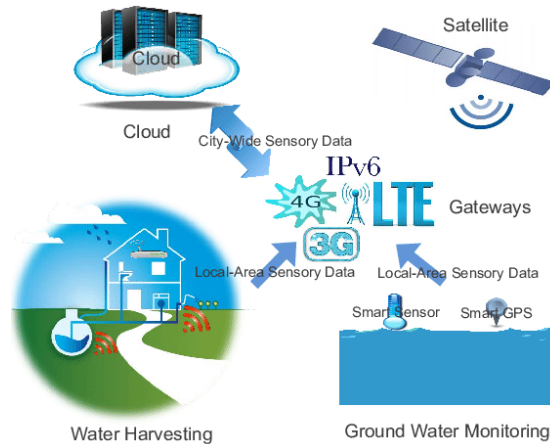


Figure 03. Smart Water Management

04 IOT tools:

4.1 The hardware side of IoT:

The hardware utilized in Internet of Things systems includes anything from control devices to dashboards, routers, and bridges, as well as servers and sensors. All are handled by these devices on a regular basis. The following are the components of IoT hardware:

1. Chips:

This is a considerably bigger category that includes all electrical and electronic devices such as microcontrollers, chips, integrated circuits, radio frequency systems, and so on.

2. Sensors:

Sensors, which are one of the most basic components of an IoT system, are divided into three categories: Power Management, Sensing, and Energy.

3. Actuators:

These devices send motion to a data gathering system, such as solenoids or comb drives, which then retrieve information based on the movements.

4. Standard devices:

Tablets, Smartphones, Switches, Routers, and other commonly used devices are considered standard devices.

4.2 The software side of IoT: The software usage in Internet of Things system includes things like threat detection, system activation, security checks, and support-specific activities, and it all helps to complete tasks like data collection, processing, storage, and evaluating instructions based on the processed data from the IoT Software. Operating systems, firmware, applications, and middleware are some of the examples of IoT software. The components of IoT software are as follows:

1. Data collection:

This step covers the fundamentals of data gathering, such as sensing, filtering, measuring, aggregating, and controlling the collected data's security. It can be done from a variety of places and then spread across devices and to a central data repository once completed.

2. Device integration:

This ensures that the IoT system's components are all well-integrated. It keeps track of all the restrictions, protocols, and applications that must be followed in order for the devices to communicate successfully.

3. Application and Process Extension:

This ensures that the IoT system's components are all well-integrated. It keeps track of all the restrictions, protocols, and applications that must be followed in order for the devices to communicate successfully.

4. Real-time analytics:

There can be automated processes that run and examine the acquired data and the processing that is done on this data for specific patterns.

05 IOT Architecture:

There is not a broadly Agreed-upon IoT architecture, Researchers have offered a variety of architectural solutions.

Three-layer and Five-layer Architectures:

A three-layer architecture [4][6] is the most basic. It was first used in the early phases of a project's research. It was made up of three layers: perception, network and application.

1. The perception layer is the *physical layer*, it is equipped with sensors for sensing and obtaining data about the environment, It detects certain physical factors or recognizes other smart items in the vicinity.
2. The network layer, Connecting other smart gadgets, Network devices, and servers is its responsibility of this department, sensor data is also transmitted and processed using its features.
3. the application layer, The user's application-specific are delivered by this layer, It outlines a number of application for the *IoT*, including smart homes, smart cities, and smart water management.

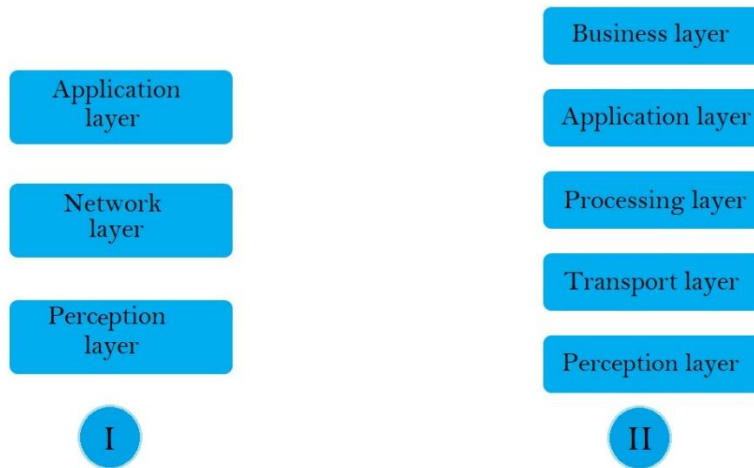


Figure04: Deferent IoT Architecture.

The three-layer design encapsulates the core concept of the Internet of Things, however it is insufficient for IoT research, because it typically concentrates on smaller details. As a result, the literature proposes many more layered architectures. The processing and business layers are included in the five-layer architecture. Perception, transport, processing, application, and business layers are the five levels that make up the system. Perception and application layers provide the same purpose as the three-layered architecture. The remaining three levels' functions are described.

1. The transport layer, Wireless, 3G, LAN, Bluetooth, RFID and NFC networks are used to send sensor data from perception layer to processing layer and the other way around.
2. The processing layer, huge volumes of data from the transport layer are stored, analyzed and processed by it, it has the ability to manage and provide a variety of services to the lower layers. Databases, cloud computing and big data processing modules are among the technologies used.
3. The business layer is in charge of the entire IoT system, including apps, business and profit models, and the privacy of users. The scope of this study does not include the business layer. As a result, we will not continue to debate it.

06 Messaging Protocols for IOT System:

6.1. COAP (**C**onstrained **A**pplication **P**rotocol):

The IETF CoRE (Constrained RESTful Environments) Working Group has developed a lightweight M2M protocol called CoAP, Request/response and resource/observe (a variation of publish/subscribe) architectures are both supported by CoAP [8]. It was created primarily to use simple proxies to communicate with HTTP and the RESTful Web, and it is unlike MQTT, does not use topics and instead uses Universal Resource Identifiers (URIs) [9].

The publisher distributes material to the URI, whereas the subscriber subscribes to the URI's specific resource. When a publisher adds new data to a URI, the URI notifies all subscribers of the new value. CoAP is a binary protocol that requires a 4-byte fixed header and short message payloads up to a maximum size that is determined by web server or programming technology [8].

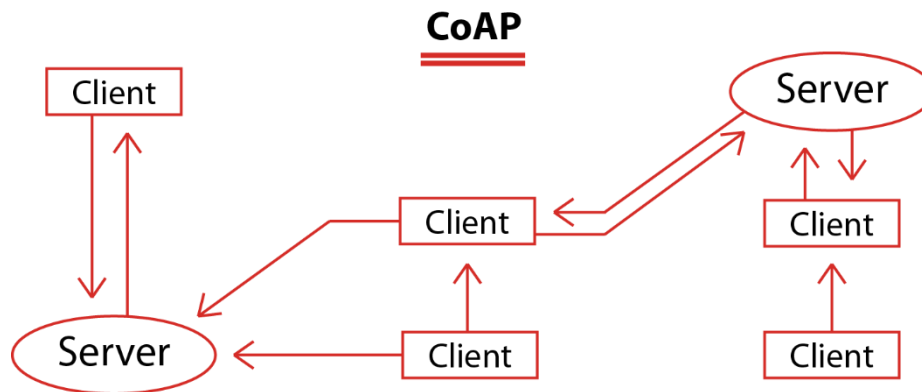


Figure 05: CoAP protocol

6.2. MQTT (**M**essage **Q**ueue **T**elemetry **T**ransport):

MQTT one of the oldest M2M communication protocols. It was created by Arlen Nipper of Arcom Control Systems Ltd and Andy Stanford-Clark of IBM (Eurotech). It's a lightweight M2M communication protocol developed for restricted networks [8]. MQTT clients send messages to a MQTT broker, which are then subscribed to by other MQTT clients or saved for future subscription. Every message is sent to a subject [10], which is an address. MQTT is a binary protocol that requires a two-byte fixed header and tiny message payloads up to 256 MB in capacity [9]. With in a transport protocol TCP and secured with TLS/SSL.

MQTT's three levels of Quality of Service (QoS) for dependable message delivery are another outstanding feature [7]. MQTT suited for vast networks of small devices that need to be monitored or controlled over the Internet by a back-end server. It is not intended for device-to-device communication or multicast data distribution to a large number of recipients [9]. It's a simple message protocol with minimal control options.

MQTT Architecture

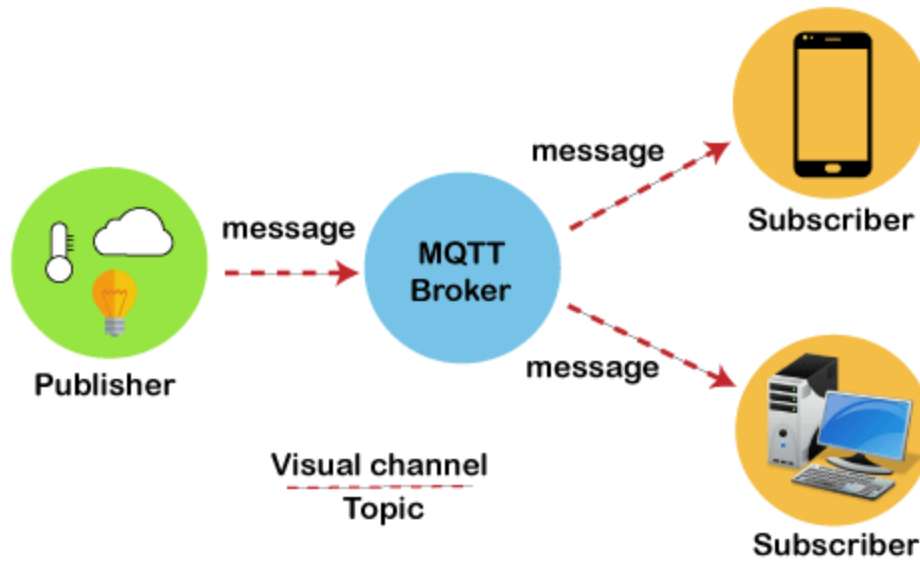


Figure 06: MQTT protocol

6.3. 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].

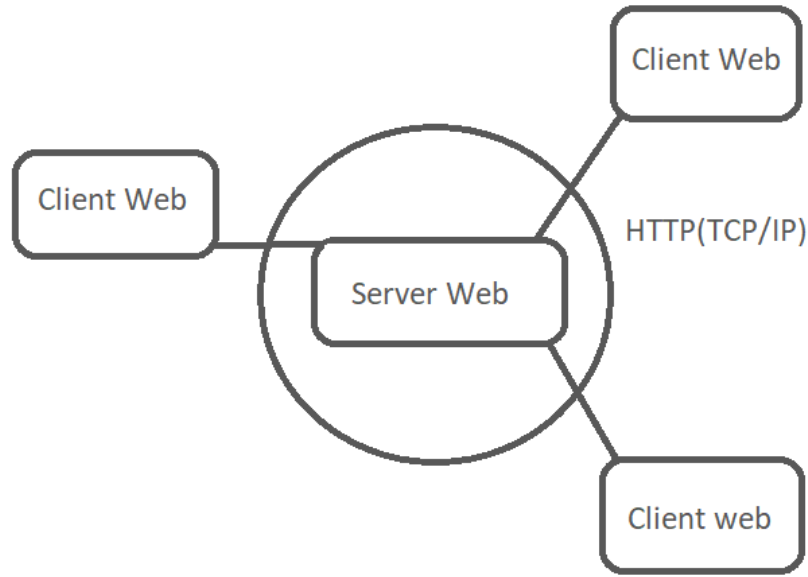


Figure 07: HTTP protocol

6.4. AMQP (*Advanced Message Queuing Protocol*):

AMQP is a lightweight M2M protocol created in 2003 by John O'Hara of JPMorgan Chase in London (UK). It is a reliable, secure, provisioning, and interoperability-focused messaging protocol [14]. Both request/response and publish/subscribe architectures are supported by AMQP [11]. It includes a variety of messaging features, including dependable queuing, topic-based publish-and-subscribe messaging, flexible routing, and transactions. AMQP's communication system requires either the publisher or the consumer to generate a "exchange" with a specific name, which is subsequently broadcasted. This exchange's name is how publishers and consumers find one another.

Upon that, a client creates a "queue" and links it to the exchange. Messages received by the exchange must be "bound" to the queue. AMQP sends and receives messages in a variety of ways, including directly, in fanout form, by subject, and based on headers. AMQP is a binary protocol that typically requires an 8-byte fixed header and short message payloads up to a maximum size that is determined by the broker/server or programming technology. With a transport protocol TCP and security provided by TLS/SSL & SASL [11]. It has two preliminary levels of Quality of Service (QoS) for reliability. As a result, Client-Broker communication is connection-oriented. AMQP's reliability is one of the most important aspects, and it provides two preliminary levels of Quality of service (QoS) for message delivery: Unsettle Format (unreliable) and Settle Format (reliable)[13].

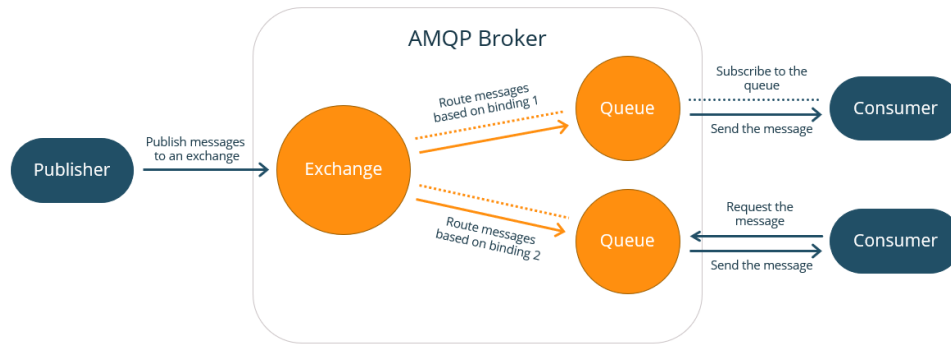


Figure08: AMQP protocol

07 Comparative Analysis of IOT Protocols:

This section compares and contrasts the four widely used and emerging messaging protocols for IOT system, COAP, MQTT, HTTP and AMQP, using numerous criteria to introduce their properties. Table 01 shows the results of the entire comparison.

TABLE 01: Comparative Analysis of Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP

Criteria	COAP	MQTT	HTTP	AMQP
Year	2010	1999	1997	2003
Architecture	Client/Server Client/Broker	Client/Broker	Client/Server	Client/Server Client/Broker
Abstraction	Request/Response(or) Public/Subscribe	Public/Subscribe	Request/Response	Request/Response(or) Public/Subscribe
Header Size	4 Byte	2 Byte	Undefined	8 byte
Message Size	Small and Undefined (normally small to fit in single IP datagram)	Small and Undefined (up to 256 MB maximum size)	Large and Undefined (depends on the web server or the programming technology)	Negotiable and undefined
Cache and Proxy Support	Yes	Partial	Yes	Yes
Semantic/Methods	Get, Post, Put, Delete	Connect, Disconnect, Publish, Subscribe, <u>Unsubscribe</u> , <u>Close</u>	Get, Post, Head, Put, Patch, Options, Connect, Delete	Consume, Deliver, Publish, Get, Select, Ack, Delete, Nack, Recover, Reject, Open, Close
Quality of service (QoS)/ Reliability	Confirmable message (similar to At once)/ Non-confirmable message (similar to At least once)	QoS 0 - At most once (Fire-and -Forget) QoS 1 - At least once QoS 2 - Exactly once	Limited (via transport Protocol - TCP)	Settle Format (similar to At most once) / Unsettle Format (similar to At least once)
Transport protocol	UDP, SCTP	TCP (MQTT-SN can use UDP)	TCP	TCP, SCTP
Security	DTLS, IPsec	TLS/SSH	TLS/SSL	TLS/SSL, IPsec, SASL
Default Port	5683 (UDP Port)/ 5684 (DLTS)	1883/ 8883 (TLS/SSL)	80/ 443 (TLS/SSL)	5671 (TLS/SSL), 5672
Encoding Format	Binary	Binary	Text	Binary
Standards	IETF, Eclipse Foundations	OASIS, Eclipse foundations	IETF and W3C	OASIS, ISO/IEC
Licensing Model	Open Source	Open Source	Free	Open Source
Organisational Support	Large Web Community Support, Cisco, Contiki, Erika, IoTivity	IBM, Facebook, Eurotech, Cisco, Red Hat, Software AG, Tibco, ITSO, M2Mi, Amazon Web Services (AWS), InduSoft, Fiorano	Global Web Protocol Standard	Microsoft, JP Morgan, Bank of America, Barclays, Goldman Sachs, Credit Suisse

08 Standards And Standardization:

As IoT devices continue to saturate society, standardization is essential to achieve universally accepted specifications and protocols for true interoperability between IoT devices and applications. Today's standards mark a major milestone for the Internet of Things by offering the unique value proposition of a single interoperability platform for all activated devices, [30]

Nearly 140 organizations around the world are now directly or indirectly affected by the standardization of M2M communication. This phase of standardization is indeed one of the crucial factors in the evolution of the mobile Internet towards the Internet of Things. There are thousands of standards "specific" to particular IoT contexts, among them and in particular those already used by industry finding those offered by the Internet Engineering Task Force (IETF), Institute of Electrical and Electronic Engineers (IEEE), International Telecommunications Union (ITU) and Global Standard1 (GS1), Organization for the Advancement of Structured Information Standards (OASIS) , as illustrated in the following table 02.

Emitter	standard	Definition
UIT	UIT-T Y.2060	Concept IoT
	UIT-T Y.2061	Interface machine-application
IEEE	IEEE 802.15.4	Data link layer
IETF	6LoWPAN	IPv6 over Low Power Wireless Personal Area Network
	CoAP	Constrained Application Protocol
	RPL	IPv6 Routing Protocol For Low-power and lossy networks
GS1	ONS	Object Naming Service
	EPC	Electronic Product code
OSIS	MQTT	Message Queue Telemetry Transport
	AMPQ	Advanced Message Queuing Protocol
	DDS	Data Diffusion Service

Table 02 : Organizations Standards de protocole IoT

- ITU: the two recommendations, ITU-T Y.2060 which provides a general view of the IoT concept and the ITU-T Y.2061 that describes the conditions for the machine interface-oriented to communications applications in the NGN environment

- IEEE and IETF in the field of IP protocol sensor networks. These efforts were first realized by the proposal of a layer model on the OSI model as well as protocols more suited to industrial networks than the TCP/IP model on Ethernet.

- GS1: has proposed the EPC (Electronic Product Code) system, which is a unique individual identifier for identifying an electronic product, as well as the EPC global Network architecture that defines the organization of information systems to ensure the exchange of EPC information at the global level. One of its main components, the ONS (Object Naming Service), is directly based on the DNS (Domain Name System).

- OASIS [34]: a not-for-profit consortium that guides development and adoption of "open" standards for the information society. The consortium's work on the Internet of Things focuses on standardized network and messaging technologies such as Message Queue Telemetry Transport (MQTT), Advanced Message Queuing Protocol (AMQP), and the Data Distribution Service (DDS). These protocols are at the application layer

09 CONCLUSION:

In the purpose of introducing and explaining our research field, we have given in this chapter a brief presentation of the Internet of Things, we have first begun by some IoT definitions, and some fields which use IoT applications, then we passed to present some technical issues as the essential IoT elements, the IoT architecture and some most famous used protocols where we focus in the application protocols. We finish by giving challenges (aspects) of IoT.

In the next chapter, we will present in detail the IoT application Protocol CoAP.

A teal-colored graphic element resembling a scroll, with a vertical strip on the left and a horizontal strip on the right. The text "CHAPTER 02" is centered on the horizontal strip in white, serif font.

CHAPTER 02

A white rectangular box with a thin black border. The text "COAP PROTOCOL" is centered inside the box in black, serif font.

COAP PROTOCOL

Chapter 02: CoAP Protocol theoretically

01 INTRODUCTION:

With their limited resources, small gadgets are unable to communicate. We must attach importance to contrast in addition to the Internet of Things (IoT).mostly because billions of sensors, computers, and other communication devices must communicate with one another, some of which may use different protocols. As a logical result, the Internet Engineering Task Force (IETF) developed the constrained Application Protocol (CoAP) to address this problem.

In this chapter, We will obtain a theoretical understanding of the CoAP protocol by looking at it is definition, features, structure, and characteristics, as well as the applications that use it, and we will compare it to IoT characteristics.

02 DEFFINITION OF CoAP:

CoAP (CONSTRAINED APPLICATION PROTOCOL) is an application layer protocol developed by the IETF CoRE Working Group, It is constructed and engineered for a constrained environments, The protocol treats the various items in the network as resources, based on a REST type design. Each resource is given a unique Universal Resource Identifier (URI). The protocol uses the corresponding URI to operate the different resources.[15]

In 2010, the IETF CoRE working group start on the development of CoAP that focus on environments of low power IP network enabling interoperability between constrained devices and the general device communication over the Internet.[16]

The Constrained Application Protocol (CoAP) is a relatively recent web transfer protocol that was created for constrained devices and networks (e.g., modest electronic devices with limited capabilities / ,e.g. LLNs, 6LoWPAN). [2]

03 FEATURES CoAP (CONSTRAINED APPLICATION PROTOCOL):[17]

CoAP has the following main features:

- Web protocol fulfilling M2M requirements in constrained environments.
- User Datagram Protocol (UDP) binding with the optional reliability, supporting unicast and multicast requests.
- Security binding to Datagram Transport Layer Security (DTLS).

- 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.
- Simple proxy and caching capabilities.
- URI and Content-type support.
- Low header overhead and parsing complexity.
- Asynchronous message exchanges.

04 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.

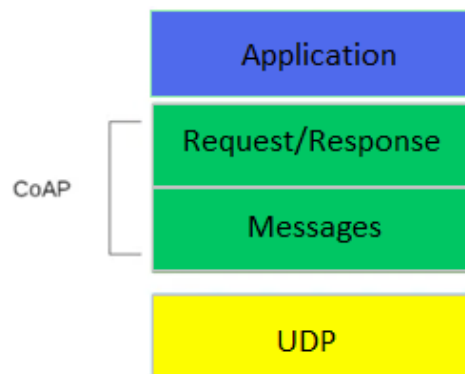


Figure 09: CoAP architecture.

4.1 Message Layer:

This is the lowest layer of CoAP. This layer deals with UDP exchanging messages between endpoints. Each CoAP message has a unique ID, this is useful to detect message duplicates. A CoAP message is built by these parts: A binary header, A compact options, Payload. [18]

Message Layer supports 4 types message: CON (confirmable), NON (non-confirmable), ACK (Acknowledgement), RST (Reset).

A communication that can be confirmed is a message that can be trusted. When messages are exchanged between two endpoints, they can be trusted. A Confirmable message is used in CoAP to obtain a dependable message (CON). The client can be confident that the message will reach the server when using this type of communication. Until the other party sends an acknowledge message, a Confirmable message is delivered repeatedly (ACK). The confirmable message's ID is repeated in the ACK message (CON). [3].

The picture below shows the message exchange process:

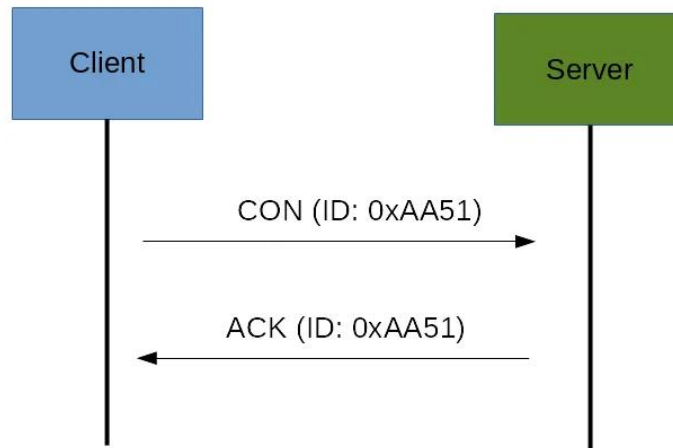


Figure 10: Reliable message transmission (CON)

If the server has troubles managing the incoming request, it can send back a Rest message (RST) instead of the Acknowledge message (ACK):

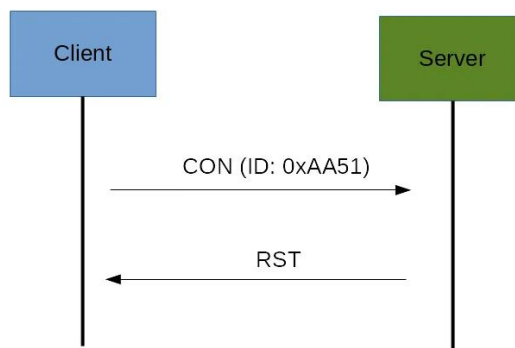


Figure 11: Reliable message transmission (RST)

4.2 Request/Response layer:

The CoAP Request/Response is the second layer in the CoAP abstraction layer. The request is sent using a Confirmable (CON) or Non-Confirmable (NON) message. There are several scenarios depending on if the server can answer immediately to the client request or the answer if not available.

4.2.1 First scenario: Piggybacked response

If the server can answer immediately to the client request, then if the request is carried using a Confirmable message (CON), the server sends back to the client an Acknowledge message containing the response or the error code:[3]

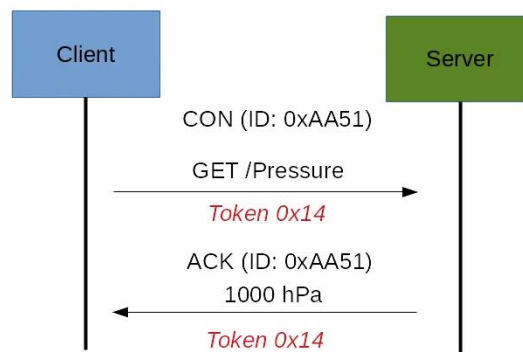


Figure 12: Piggy-backed request/response transmission

There is a Token in the CoAP message, as you can see. The Token is not the same as the Message-ID and is used to match the request and response.

4.2.2 Second scenario: Separate response

If the server is unable to respond to the client's request right away, it sends an Acknowledge message with an empty response. The server sends a new Confirmable message to the client containing the response as soon as it is available. The client now responds with an Acknowledge message:[4]

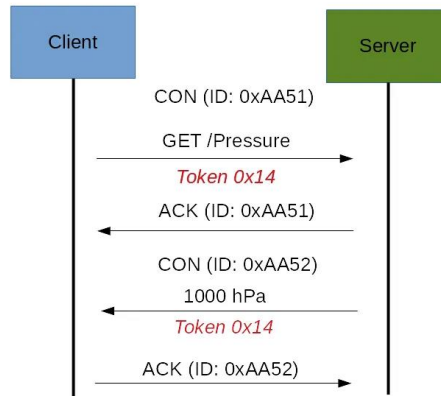


Figure 13: Separate request/response transmission

4.2.3 Third scenario:

The other message category is the Non-confirmable (NON) messages. These are messages that don't require an Acknowledge by the server. They are unreliable messages or in other words messages that do not contain critical information that must be delivered to the server. To this category belongs messages that contain values read from sensors. Even if these messages are unreliable, they have a unique ID.

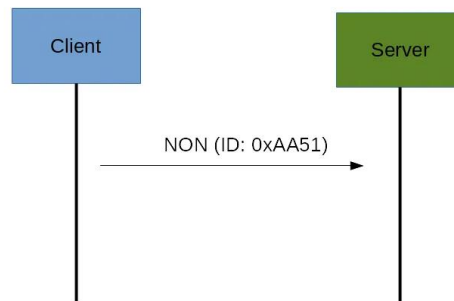


Figure 14: Reliable message transmission

4.2.4 CoAP request methods:

CoAP supports the basic methods of GET, POST, PUT, DELETE, which is easily mapped to HTTP.

CoAP operations are safe (only retrieval) and idempotent (may be invoked numerous times with the same consequences) since they manipulate resources. Because the GET method is safe, it must not perform any additional actions on a resource beyond retrieval. The GET, PUT, and DELETE procedures MUST all be done in an idempotent manner. Because the URI in the request indicates the resource that will handle the contained body, POST is

not idempotent like PUT. As a result of the POST, this resource identified by the POST may be utilized for data processing, as a gateway to other protocols, or to build a new resource.

Delete: The DELETE method requests that the resource identified by the request URI be deleted. The response 200 (OK) SHOULD be sent on success. Responses to this method are not cacheable.

Put: The PUT method requests that the resource identified by the request URI be updated or created with the enclosed message body. If a resource exists at that URI the message-body SHOULD be considered a modified version of that resource and a 200 (OK) response SHOULD be returned. If no resource exists then the server MAY create a new resource with that URI, resulting in a 201 (Created) response. If the resource could not be created or modified, then an appropriate error response code SHOULD be sent. Responses to this method are not cacheable.

Post: The POST method is used to request the server to create a new subordinate resource under the requested parent URI. If a resource has been created on the server, the response SHOULD be 201 (Created) including the URI of the new resource in a Location Option with any possible status in the message body. If the POST succeeds but does not result in a new resource being created on the server, a 200 (OK) response code SHOULD be returned. Responses to this method are not cacheable.

Get: The GET method retrieves the information of the resource identified by the request URI. Upon success a 200 (OK) response SHOULD be sent. The response to a GET is cacheable if it meets the requirements in caching.

Code Range	Type	Code	Name / Description
0.01-0.31	Request Methods	0.01	GET
		0.02	POST
		0.03	PUT
		0.04	DELETE

Table 02: CoAP request codes

4.2.4 CoAP request methods:

Response codes are similar to HTTP. For instance, 2.xx indicates success, 4.xx indicates client error and 5.xx indicates a server error. Although some response codes match the HTTP status codes (e.g., 4.04 and 404 [Not Found]), others have different codes (2.05 [Content] is equivalent to 200 (OK), but 2.05 is only used in response to GET) or are not represented at all.

Code Range	Type	Code	Name / Description	
2.00-5.31	Success Response	2.01	Created	
		2.02	Deleted	
		2.03	Valid	
		2.04	Changed	
		2.05	Content	
	Reserved		3.0-3.31	Reserved
	Client Response	Error	4.00	Bad Request
			4.01	Unauthorized
			4.02	Bad Option
			4.03	Forbidden
			4.04	Not Found
			4.05	Method Not Allowed
			4.06	Not Acceptable
			4.12	Precondition Failed
	Server Response	Error	4.13	Request Entity Too Large
4.15			Unsupported Content-Format	
5.00			Internal Server Error	
5.01			Not Implemented	
5.02			Bad Gateway	
6.00-7.31	Reserved	5.03	Service Unavailable	
		5.04	Gateway Timeout	
		5.05	Proxving Not Supported	

Table 03: CoAP respond codes.

05 CoAP MESSAGES FORMAT :

In this paragraph, we present the CoAP message format. The constrained application protocol is fundamental for restricted environments, and for this reason, it uses built-in messaging. To shun fragmentation, the message occupies the data section in the UDP datagram.

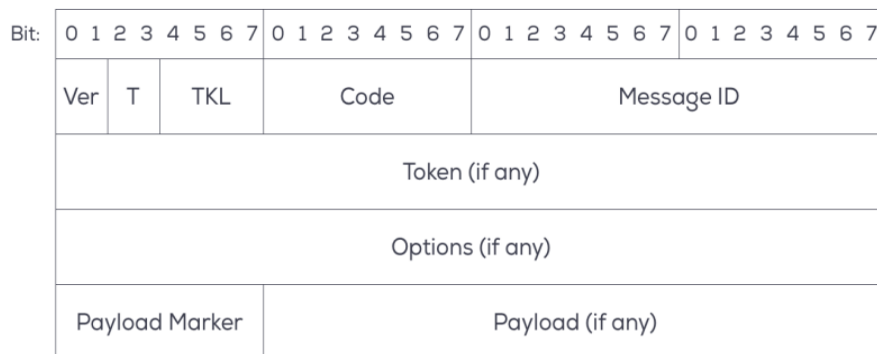


Figure 15: CoAP message format

The CoAP header has been designed to be easy to analyze by programs running on small devices such as sensors (Figure 00 up), The CoAP header begins with a fixed four-byte portion, which includes:

Protocol version (Ver): this is the version of the CoAP protocol, namely 1 (2 bits)

The type of message (T): Indicates that the message is of type Confirmable (0), Non-confirmable (1), Acknowledgment (2), or Reset (3). (2bits)

Token length (TKL): Indicates the length (variable) of the Token field (0-8 bytes).

Req/Resp Code: 8-bit, 3-bits class (the most important bits) and 5-bits detail (the least significant bits) are separated. "C" is a digit between 0 and 7 (3 bits) and "DD" is two digits between 00 and 31 (5 bits), it described in the form C.DD. The class can indicate a request (0), a success response (2), a client error response (4), or a server error response (5). All other class values are reserved. A special case is that of an empty message whose code is 0.00. In the case of a request, the Code field indicates the method.

Methods	Codes
GET	0.01
POST	0.02
PUT	0.03
DELETE	0.04

Figure 16: methods and codes

Message ID: a 16-bit used for authoritative transmission, A maximum of 250 messages per second are allowed. This isn't a significant limitation: CoAP machines' resources don not allow them to be overly communicative in the first place.

Token: Unauthenticated integer of 4 bytes. The length of the variable-length Token value, which can be between 0 and 8 bytes long [4]. A request and its response are linked using the token value.

Option (TLV frame): The option number, option value length, and the value itself must all be included in the options field, if one exists. The option number is not declared specifically. Rather, the equation "option number = option delta + prior option number" is used to calculate it.

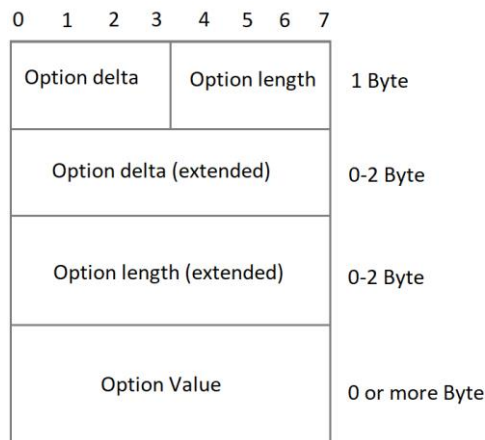


Figure 17: Option Format

Option delta: is used to determine the difference between the current option number and the one before it.

Option length: Simply specify the option value's size.

Option value: A sequence of exactly Option Length bytes. The length and format of the Option Value depend on the respective option, which MAY define variable-length values. for the formats used in this document; options defined in other document MAY make use of other option value formats.

CoAP defines several options used on both request and responses:

To specify the target resource of a request to a CoAP source server, utilize the Uri-Host, Uri-Port, Uri-Path, and Uri-Query options. The options encode the various components of a request URI in such a way that the option values show no percent-encoding and the whole URI may be rebuilt at any associated endpoint. Each option holds the following values:

Uri-Host: the Uri-Host Option specifies the Internet host of the resource being requested

Uri-Port: the Uri-Port Option specifies the transport-layer port number of the resource

Uri-Path and Uri-Query: each Uri-Path Option specifies one segment of the absolute path to the resource, and each Uri-Query Option specifies one argument parameterizing the resource.

Proxy-Uri: The forward-proxy is asked to transmit the request or service it from a legitimate cache and reply, A request is forwarded to a forward proxy.

Proxy-Scheme: In a proxy request, it's utilized to pull together an absolute-URI. The absolute-URI is made up of the choices Uri-Host, Uri-Port, Uri-Path, and Uri-Query.

Content-Format: The Content-Format Option specifies the message payload's representation format. The representation format is specified using a numeric Content-Format identifier from the "CoAP Content-Formats" registry.

Accept: The CoAP Accept option can be used to provide the client's preferred Content-Format. The representation format is specified using a numeric Content-Format identifier from the "CoAP Content-Formats" registry.

Max-Age: indicates a maximum time in which the response cached can be considered fresh.

ETag: An entity-tag is intended for use as a resource-local identifier for differentiating between representations of the same resource that vary over time.

Location-Path and Location-Query: defines an absolute path, a query string or both that specifies the new location and/or query argument of a resource created with a POST request.

Conditional Request Options: Conditional request options allow a client to request that the server only execute the request if certain circumstances indicated by the option are fulfilled.

Size1 Option: The Size1 option provides size information about the resource representation in a request. The option value is an integer number of bytes. Its main use is with block-wise transfers

if-match: The If-Match Option MAY be used to make a request conditional on the current existence or value of an ETag for one or more representations of the target resource.

if-non-match: The If-None-Match Option MAY be used to make a request conditional on the nonexistence of the target resource. If-None-Match is useful for resource creation requests, such as PUT requests.

The payload: The Payload Maker is a set of one byte in the payload that signals the beginning of the payload data. There is data present if the Payload Maker has a value of all ones (0xFF16), otherwise, the payload is empty.

06 COAP PROTOCOL CHARACTERISTICS:

6.1 Caching:

Endpoints that support CoAP MAY store results in order to minimize response time and network bandwidth consumption for similar requests in the future.

In CoAP, the purpose of caching is to reuse a previous response message to fulfill a current request. A "freshness" method is utilized in some circumstances to reuse a stored response without the requirement for a network request, decreasing latency and network round-trips. Even when a fresh request is necessary, it is frequently possible to reuse the payload of a previous answer to satisfy the request, minimizing network bandwidth utilization. This is accomplished through the use of a "validation" method.[4]

Unlike HTTP, the cacheability of CoAP response is determined by the Response Code rather than the request method. Each Response Code's cacheability is determined by the Response Code specifications. Caching of response codes that signal success but are not acknowledged by the endpoint is not recommended.[4]

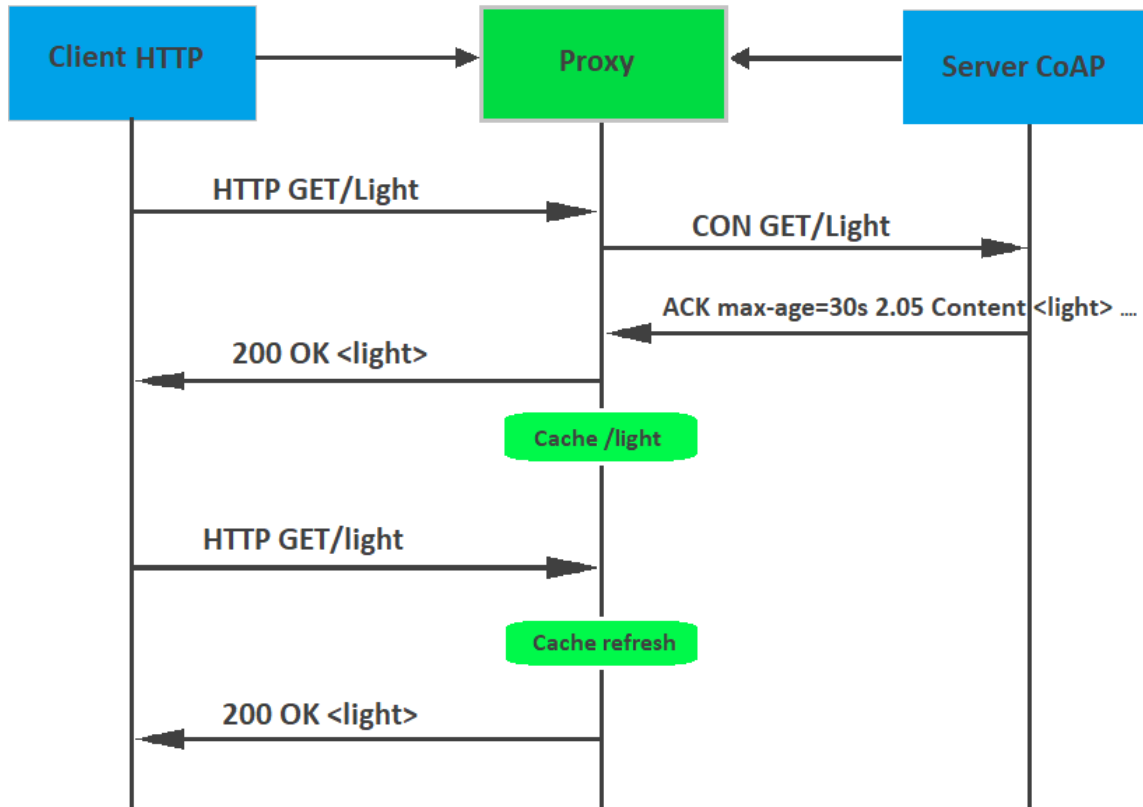


Figure 18: Caching in CoAP.

6.2 Proxying:[26]

The COAP proxy is intended to support applications that must communicate with WSN nodes, such as smart house development.

The network architecture, protocol stack, and proxy role are depicted in figures 19 and 20.

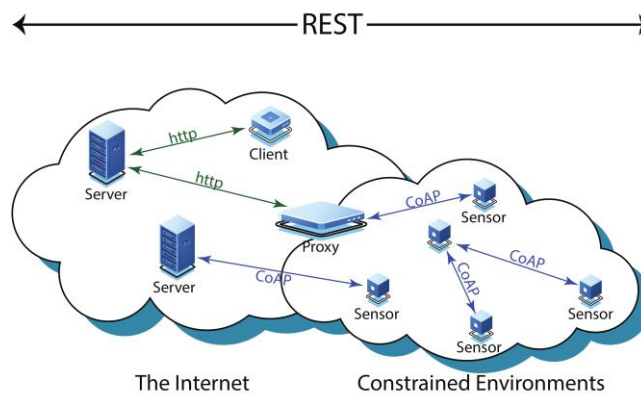


Figure 19: Network Architecture Using Proxy

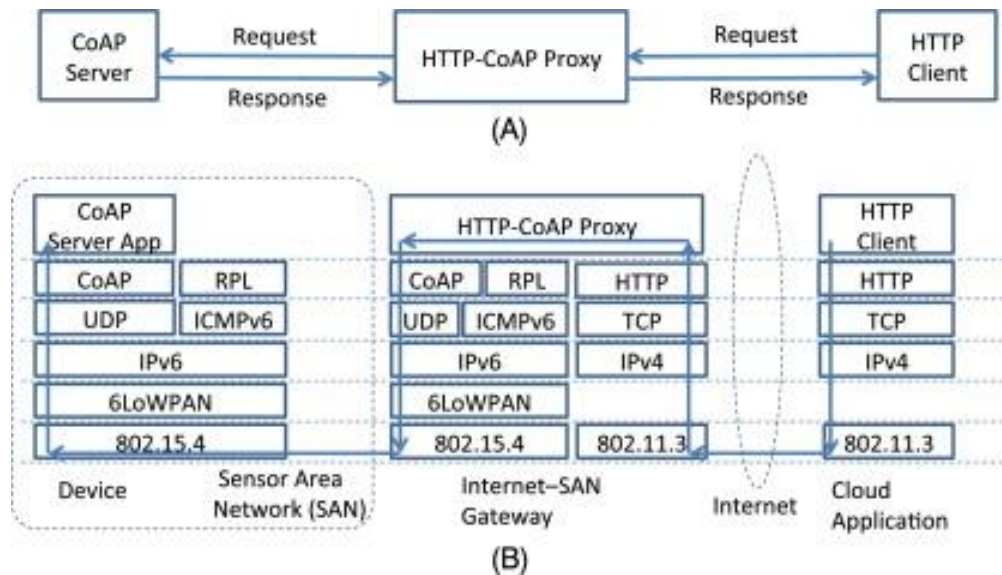


Figure 20: Protocol Stack

A proxy is a CoAP endpoint that CoAP clients can use to make requests on their behalf. This might be advantageous if the request could not be made otherwise, or if the response is served from a cache to decrease response time, network bandwidth, or energy usage. To link disconnected CoAP networks, the CoAP proxy additionally offers IPv6 over Low Power Wireless Personal Area Networks (6LoWPAN) edge router and gateway features.[3]

The similarities between CoAP and HTTP protocol enable a straightforward mechanism to construct proxy between these two protocols. There are two sorts of proxies on a CoRE structure: forward-proxy, selected by a client, and reverse-proxy, picked by the origin server. As a result, two devices that implement the two protocols may be readily linked via the Proxy. Cross-protocol proxying can be divided into two categories:

Proxying CoAP-HTTP: Since CoAP clients Allowed to use an intermediary to access resources on HTTP servers. This is started by providing the Proxy-Uri or Proxy-Scheme Option in a CoAP request to a CoAP-HTTP proxy with a "http" or "https" URI.

Proxying HTTP-CoAP: It Allows HTTP clients to use an intermediary to access resources on CoAP servers. An HTTP request to an HTTP-CoAP proxy is started by supplying a "coap" or "coaps" URI in the Request-Line.

6.3 Resource Director:[19]

The "well-known/core" URI is appropriate for resource discovery only when the discovery is internal to the WSN. When a device or program outside the WSN needs to learn about the resources housed in its nodes, it becomes inefficient. This process is made easier by using an RD object. The description of the resources given by WSN nodes is stored in the RD. The resource description in CoAP is done using the Core link format, which is similar to HTTP's Web linking. WSN nodes are responsible for registering and maintaining their RD entries. The RD has APIs for registering, updating, and removing items, as well as resource discovery. The "well-known/core"-URI is the RD's entry point.

The client in the example below is requesting a list of the server's available resources (GET /.well-known/core). The returned list (in CoRE Link Format) reveals that the server has a resource named /s/t that provides the temperature in degrees Celsius when asked. The client then requests the value of this resource (GET /s/t), and the server responds with a plain text message with the value of the actual temperature as payload.

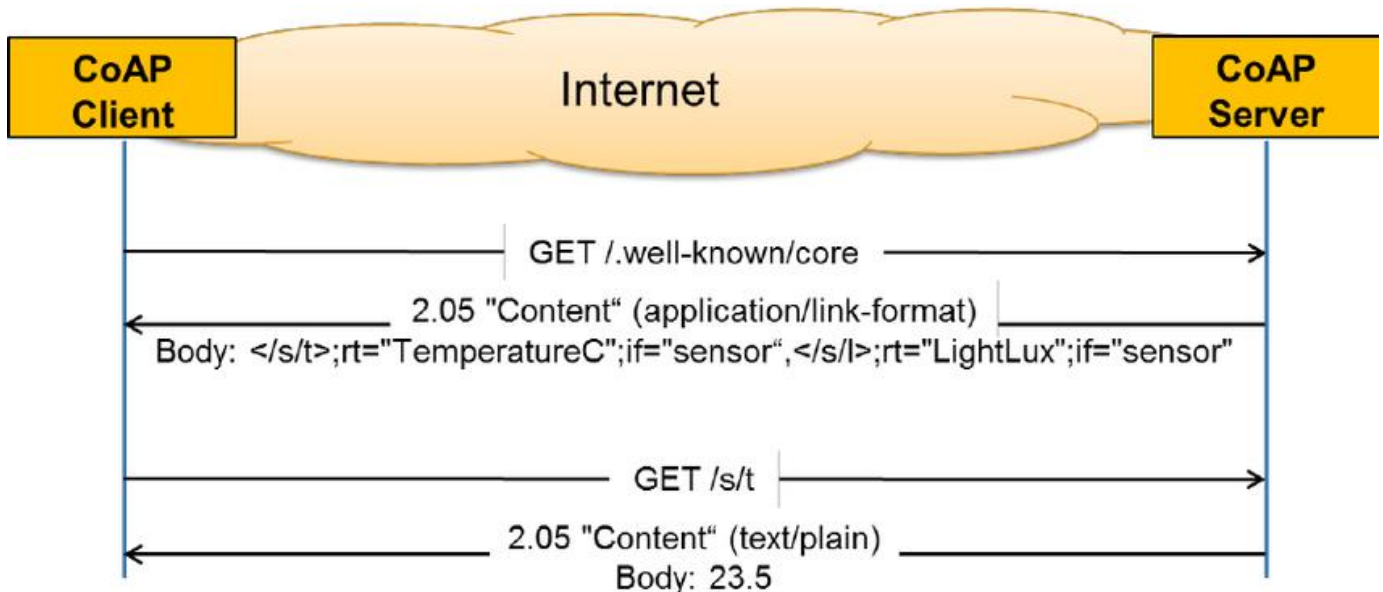


Figure 21: An example of Constrained RESTful Environments (CoRE) direct resource discovery and (CoAP) request.

Direct resource discovery, on the other hand, is difficult in many M2M environments because nodes may slumber for long periods of time. CoRE Resource

Directories (RD), which host descriptions for resources on other servers, can be used to overcome this problem. The CoAP server can then register its resources with one or more RDs in this manner. Clients can find these resources by conducting searches against RD.

For example, the same resource discovery that was done in upper Figure utilizing direct communication between the client and the server can now be done in lower Figure using an RD.

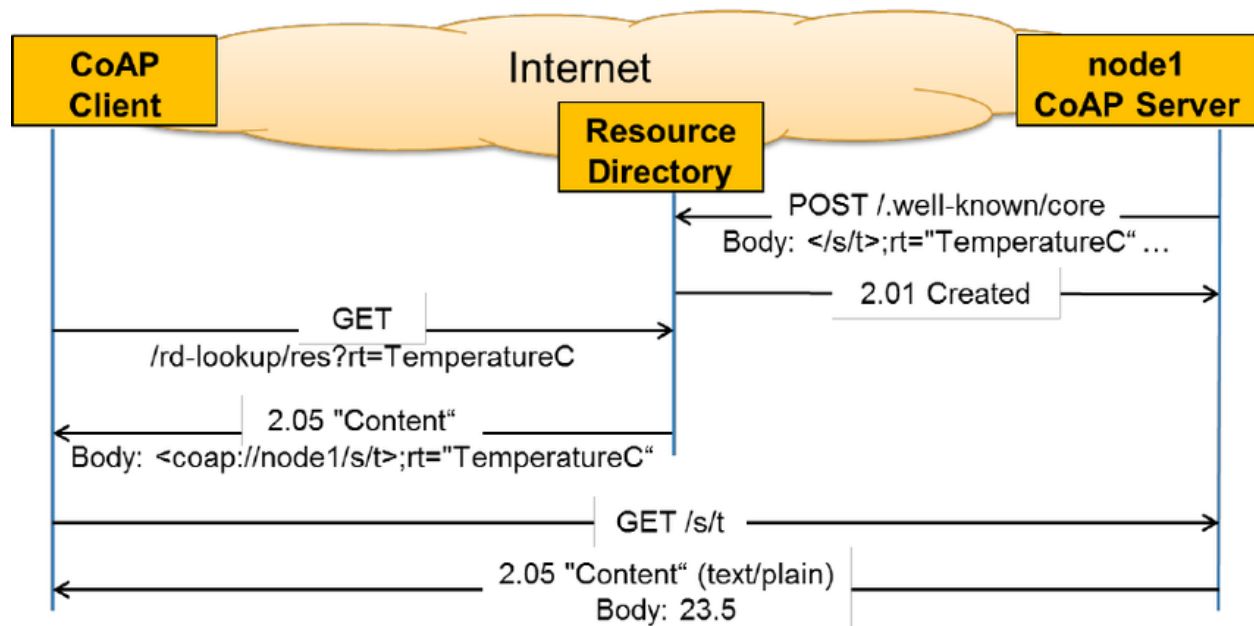


Figure 22: An example resource discovery by using a Resource Directory.

07 EXAMPLE OF EXCHANGES IN CoAP:

First process: The client sends a Confirmable (CON) GET request for the resource CoAP://server/pressure to the server with a Message ID of 0xAA51. The request includes one Uri-Path Option (Delta 0 + 8 = 8, Length 8, Value " Pressure ").

Second process: A 2.05 (Content) response is returned in the Acknowledgement (ACK) message that acknowledges the Confirmable request, repeated both the Message ID 0xAA51. The response includes a Payload of "1000 hPa" and is 11 bytes long.

Third process: The client sends a GET request with a message ID 0x7d35, of confirmable type CON, and with a Uri-query = / humidity.

Fourth process: The Confirmable GET request is no longer reachable. The client retransmits the request after ACK TIMEOUT seconds.

Fifth process: The server acknowledges the Confirmable request by sending a 2.05 (Content) response to the request as payload=60.0 and a type ACK acknowledgment.

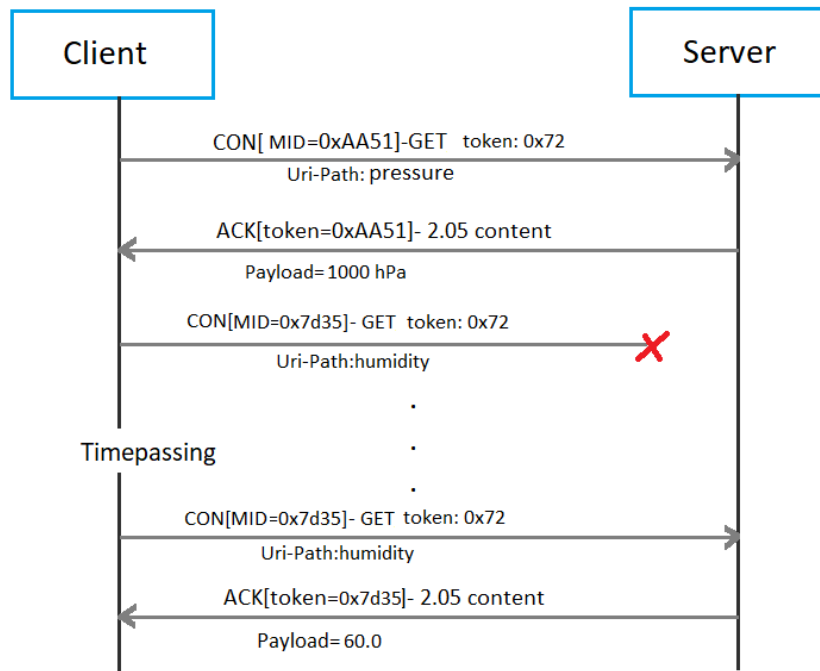


Figure 23: CoAP example communication

08 FUNCTIONAL APPLICATIONS OF CoAP PROTOCOL:

Many locations have successfully incorporated CoAP, including:

Health Domain and Comfort	Aliveness In autonomous	help for the elderly / help for the disabled
		people at home mobility assistance
Industry Domain	Industrial processing	monitoring of industrial installations
		baggage handling - boarding operation
		real-time vehicle diagnostics - assembly process
Smart City Domain	Agriculture And Livestock	farm registration manager
		monitoring irrigation, production Agricole
	Smart Energy	load management - storage service - entertainment services
		sustainable mobility
		energy: production - distribution - storage - management
	smart travel and tourism	road condition monitoring
		traffic management

Table 04: FUNCTIONAL APPLICATIONS OF CoAP PROTOCOL

09 SECURING CoAP:

The security of CoAP is an important aspect due to not having the reliable standards to secure CoAP architecture. For this purpose, DTLS is introduced. Integrity, authentication, confidentiality, and non-repudiation services, are all aspects of security. All of them are managed to meet by DTLS.

End-to-end communication is protected by the it in the application layer protocol. As a result, end-to-end communication makes it more difficult for the attackers to gain access to the data. Also, It solves issues like packet loss and reordering.

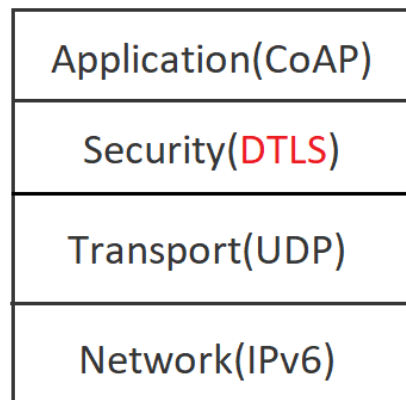


Figure 24: Abstract Layering of DTLS-Secured CoAP

DTLS controls critical security features such as message authentication, secrecy, key management, and data integrity. It has four security modes that CoAP uses with a variety of applications, as mentioned below:

NoSec Mode: This mode expects that the security mechanism will be implemented by another protocol layer, hence messages are sent with no security.

Pre-Shared-Key Mode: Devices that are permitted to utilize the same system are pre-programmed with a single symmetric key that helps communication with other devices.

Raw-Public-Key Mode: it is regarded to be critical in the implementation of CoAP and is commonly used by devices that require authentication. It includes an asymmetric key for each device to assist in identifying and interacting with it.

Certificates Mode: This is an authentication approach for devices that use an X.509 certificate to implement CoAP.

10 CoAP AND IOTASPACT:

10.1 Energy Consumption:

Because many of the devices that consist IoT environments lack constant sources of energy and tend to depend on batteries or some form of energy harvesting, one of the many challenges that IoT devices face is providing an efficient system of protocols that minimizes power consumption while maintaining the same level of performance and capabilities as any other Internet node.

The availability of useful power in a constrained node network varies. On constrained environments, at least one of the following energy limitation types will exist:

- **Occasions and Events:** define limits of the power amount used for a every single event, such as pressing a button.
- **Duration:** have access to maintenance, such as having their main battery replaced, or have the capabilities to collect power.
- **Continuance:** Because there is no other way to recharge or replace its main battery, it must have a certain amount of power available throughout its lifetime.

The Physical and MAC layers are more concerned with power management than the Network, Transport, and Application layers. With the rise of Internet-connected smart objects, some changes were required to better reduce energy consumption. The dearth of an unlimited source of energy supply by constrained devices is taken into account by CoAP, for example. It also includes a few implementations that can help reduce the amount of energy used in data transmission and processing. Proxy servers and caching can be used to reduce response times and energy consumption. Furthermore, the simplicity of UDP and the descriptive but short URIs reduce processing time, lowering energy consumption.

10.2 Infrastructure communication

A protocol's appropriateness is heavily dependent on the network environment and nodes existing in that it. Analyzing the type of network, its circumstances, limitations, and requirements to be satisfied might logically impact the protocol's overall performance, especially if all or most demands are treated correctly. As previously indicated, CoAP thrives in low-power, lossy networks, where devices are 8-bit microcontrollers with limited ROM and RAM, substantial packet rate loss, and

bandwidth of 10 Kbit/s. CoAP appears to be acceptable for use on short distance (less than a mile) networks and is suitable for both 6LoWPAN and LLN networks.

Integration with existing Web infrastructure is considerably easier with CoAP because it is built on REST architecture and shares method and response codes. Because both HTTP and CoAP are REST-based, IoT and Internet devices can simply use cross-proxy to transfer the request/response model from one standard to another, avoiding the complexity of application gateway implementations.

CoAP is a modern protocol that optimizes RESTful architecture for M2M apps and supports and adapts various HTTP capabilities for restricted devices. Because CoAP is not yet fully mature as a standard and is still relatively new, other protocols that are more well-known and frequently used on the Internet may take precedence over it. However, for a relatively new protocol, CoAP is rapidly growing rapidly with businesses, CoAP seems to have a future as a key protocol in the development of IoT.

10.3 Reliability:

With the adoption of UDP as the primary transfer protocol, CoAP achieves reliability through Confirmable messages that expect an Acknowledge in return. Even so, while this mechanism ensures that a message reaches the intended recipient, it provides no indication that the message was delivered successfully and without errors. As a result, its dependability is minimal and optional. Because reliability is optional, some messages may be marked as Non-confirmable and do not require acknowledgment, However, both types of messages use Message ID to avoid duplicates.

10.4 Scalability:

CoAP has a number of characteristics that set it apart from other protocols in terms of processing power, probably that making it a better suit for IoT. Device, service, and resource discovery are all supported. The discovery mechanism used by the CoRE connection format is versatile, allowing for device mobility and scalability. Nodes can perform an automatic activity that allows them to locate other devices and resources, making it easier to replace or add nodes to the network. This capability is especially useful in vast areas with a large number of endpoints.

10.5 Compatibility:

Internet protocol standards is still under progress, particularly for IoT devices. The IETF CoRE working group has made and continues to make significant efforts to standardize CoAP. In IoT ecosystems, the standardizing of Internet protocols is crucial for compatibility between numerous diverse IoT devices, applications, and networks. It is a critical feature for a protocol's long-term survival on the WoT.

Because CoAP is an free open-source web protocol, its source code is freely available for anybody to access, change, and redistribute. The key benefit of having this protocol open-source is that it can address its maturing issue by encouraging IoT developers and industry to experiment with it, resulting in its acceptance. However, one disadvantage of open-source is that it might lead to numerous different implementations of the protocol, making interoperability impossible. In a similar way, an open-source protocol advantage through code transparency, which allows for changes in the code to support compatibility.

10.6 Performances:

Traditional request/response models, such as those used by CoAP and HTTP, are based on regular resource polling, in which clients request resources and the server answers. Polling, on the other hand, might be harmful on a restricted network if a client requires current resource representation on a regular basis. Clients send frequent request messages to the server in order to get an updated resource in HTTP, which is solved by repeated polling, or lengthy polling. In turn, the server responds only when the new version is available. For long-lived devices, this mechanism reduces latencies and optimizes processing and network resources. However, because limited devices have limited build, that's not the preferred technique. This is particularly problematic for a CoAP proxy that talks with both CoAP devices and Web applications. Although numerous Web applications trying to receive data from the restricted network might create an increase in overhead, latency, and network traffic, the CoAP proxy may allow HTTP long polling. Overall, the increased complexity and overhead might reduce performance marginally.

As a result, non-confirmable message transmission might cause the network to overload. CoAP, as any web protocol, uses congestion management to keep the network stable and restrict the volume of messages passing through the network and nodes. Despite this, the CoAP congestion management concept is straightforward, taking just confirmable messages into account. This protocol flaw will be addressed in the future,

as evidenced by proposals for advanced congestion control, which attempt to optimize techniques for improved performance.

10.7 Effectiveness charge:

The key benefits of deploying standardized protocol solutions, such as CoAP, to minimize network costs are that non-standardized solutions would violate the end-to-end Internet principle. Using the end-to-end approach, the last few meters would require translation from a standardized Internet protocol to a customized protocol. Thus, application gateways are introduced, which not only adds to the network's complexity but also takes time and money to manage, install, and run.

Another advantage is that CoAP uses the CoRE link service/resource discovery. This is beneficial because it makes it easier to add new devices, replace old ones, and expand at a low cost since devices can use resource and troubleshooting to gather all the data they need to connect to the network, which means device maintenance is less costly because no installation or manual configuration is required.

11 CONCLUSION:

The goal of this chapter is to give a theoretical analysis of the CoAP protocol, including its features, structure, and characteristics, as well as a theoretical comparison of the protocol to the characteristics of the Internet of Things.

The simulation of the CoAP Protocol will be shown in the following chapter.



CHAPTER 03

COAP PROTOCOL SIMULATION

Chapter 03: CoAP Protocol Simulation

01 INTRODUCTION:

The simulation includes modeling the entire researched system and numerically simulating it with surroundings derived from real-world observations or probabilistic models. The ability to work on systems that aren't available is one of the benefits of simulation. It is much less expensive to run a preliminary simulation of the alternatives under consideration during the design stage, for example. Furthermore, simulation is a very adaptable method of researching an issue. This technique enables program repeats with parameter modifications and execution trace capture without the unpredictability of a real-world context.

In this chapter, we will go over some of the assessment methodologies, then go over how to simulate the CoAP protocol using the proposed scenario (movement of prisoners inside the prison), as well as the tools that were used. Using the mobility tool to added a movement experience and modifying and increasing the number of servers, we will run several CoAP experiment simulations.

RELATED WORK:

Due to its efficacy and less degree of use in the thies years, many works have studied the performance of application CoAP protocol with mobility in the IoT environment. We now review some relevant of these works in chronological order.

- In this article [1], we propose a mobility management protocol, named CoMP, which can effectively retrieve the sensing data of sensor nodes while they are moving. The salient feature of CoMP is that it makes use of the IETF CoAP protocol for mobility management, instead of using Mobile IP. Thus CoMP can eliminates the additional signaling overhead of Mobile IP, provides reliable mobility management

02 VALIDATION METHODS:

There are several methods for assessing a system's performance on a WSN. Analytical modeling, measurements based on real-life experiences, and simulation are only a few examples.

2.1 Analytical methods:

It presents analytical methods for studying a system's behavior by solving the mathematical equations that support its mathematical model. Analytical approaches are most

useful for solving equations that are relatively inexpensive to compute. Furthermore, analytical methods allow for a better knowledge of how a system works because they allow one to evaluate some of the system's imbalances by solving his model and thereby providing solutions, such as Formal approaches. In scientific writing, the analytical technique necessitates a proof, exemplary inspection, and quantitative measures.

2.2 Real Experience:

Validation of protocols and applications based on real-world testing is difficult to implement. In fact, studying an issue from real-life experiences is challenging for various reasons: Experiments can be difficult to duplicate. The study of increase, decrease, and fluctuation in speed and pattern of movement is a complex process that can be disrupted by external circumstances over which the experimenter has no control. Real experience approaches have the problem of requiring restricted hypotheses about the practical system in order to obtain effective models, and as our subjects do not require real application because this is performance research, this method was not chosen.

2.3 Simulation:

A discrete event simulation involves examining a specific impression of a system's model and duplicating its behavior. The benefit of simulation is that it provides a highly broad method that allows any model to be studied as long as the simulation tool adapts to the model being studied. On the other hand, it has the drawback of necessitating a significant amount of machine calculation time.

There are various different event simulators available. Let us mention the NS-2 and NS-3 network simulators, as well as OMNeT, which allow simulation of various types of networks, including wait networks, OPNET, and COOJA, a performance analysis tool.

The default network simulator for Contiki, COOJA, was pre-installed with Contiki 3.0. COOJA provides a user-friendly interface that enables quick simulation setup and analysis. Because our topic needs a performance analysis, COOJA was identified as one of the finest tools for simulating the protocol due to its flexibility, extensibility, and speed of prototyping.

03 METHODOLOGY:

To evaluate the effectiveness of the CoAP protocol, we will use a scenario in which inmates are moving around inside the prison. Using the COOJA simulator and the Libcoap library, we simulate various scenarios by adjusting variables. We updated several files in the Contiki system (power trace, er-coap-engine, etc...), and we constructed a coap-client, and we will retrieve the findings and turn them into curves using Python to make evaluation and analysis easier.

04 TOOLS USED IN THIS SIMULATION:

To simulate the COAP protocol, we must first understand the tools that are utilized.

4.1 Software:

✓ VMware Workstation:

VMware Infrastructure is a comprehensive infrastructure virtualization package that combines virtualization, management, resource optimization, application availability, and operational automation. VMware Infrastructure virtualizes and combines the underlying physical hardware resources across several systems, delivering virtual resource pools to the datacenter in a virtual environment.

✓ Contiki-OS: [20]

Contiki is an open-source operating system for tiny low-power microcontrollers that allows developers to create programs that make efficient use of the hardware while also providing standardized low-power wireless connectivity for a variety of hardware platforms.

It is utilized in a wide range of commercial and non-commercial systems, including city sound monitoring, street lights, networked electrical power meters, industrial monitoring, radiation monitoring, building site monitoring, alarm systems, and remote house monitoring.

✓ COOJA:

It's a versatile Java-based simulator that lets you write application software in C language utilizing the Java Native Interface. One of the major benefits of our COOJA simulator is that it can simulate application software in both high-level algorithm development and low-level hard driver development at the same time. The COOJA simulator is extremely adaptable., or current parts can be modified. We may run a variation simulation with varied conditions and system parameters, such as different packet creation rates, different MAC protocols, and different network topologies, using COOJA's advantages. [20]

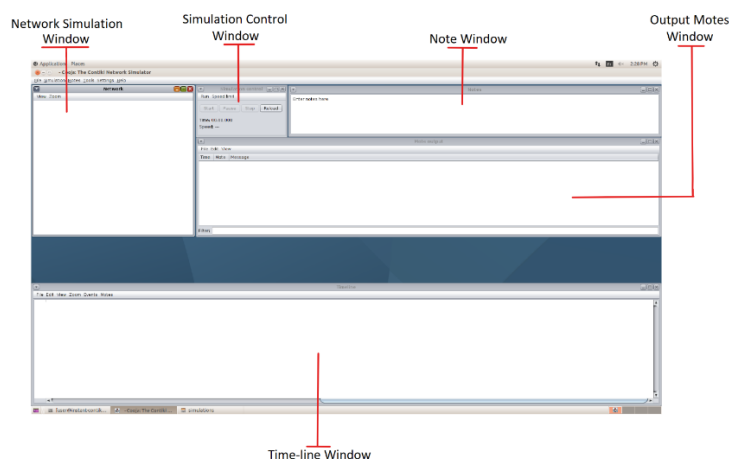


Figure 25: COOJA simulator Interface

Simulation window:

We have numerous windows in a simulation:

- *The Network window*: at the top left of the screen, shows us all the nodes in the simulated network.
- *The Simulation control window*: is where we can start, pause and load our simulation.
- *The Mote Output window*: on the right side of the screen, shows us all the serial port prints from all nodes and all exchanges messages enter the motes.
- *The Notes window*: at the top right is where we can put notes for our simulation
- *The Timeline window*: at the bottom of the screen, displays all the events of communication in the simulation over time, very convenient to understand what is happening in the network.

✓ LibCoAP library:[21]

libcoap library is a C implementation of a lightweight application-protocol for devices with limited resources such as computational power, radio frequency range, memory, bandwidth, or network packet sizes. The IETF has standardized this protocol as RFC 7252.

4.2 Programming Languages Usage:

✓ C Language:[22]

Is a compiled language (as opposed to interpreted languages). A C program language is written as a text file, referred to as a source file. Because this file cannot be executed by the microprocessor, it must be converted into machine language. A program called a compiler [new2] performs this operation. It was utilized in this project to customize the COOJA emulator's files to meet the study's criteria.



Python language:[27] Python is a programming language that was created in 1989 and is similar to C, C++, FORTRAN, Java, and others. It has the following main characteristics: open source is free to use, source files are available and editable; it comes with a large base library and a large number of libraries for scientific computing, statistics, and visualization..., and dynamic writing is done automatically during program execution, allowing for greater programming flexibility and speed, but it is motivated by excessive memory consumption and performance loss, allowing for [23]. It was used in this project to create curves based on the findings.



- ✓ JAVA language: Java is a programming language and computing platform first released by Sun Microsystems in 1995. There are lots of applications and websites that will not work unless you have Java installed, and more are created every day. Java is fast, secure, and reliable. From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere. [29]



4.3 Hardware:

The computer used to simulate the study has the following characteristics:

- ✓ Computer: Asus FX503VM
- ✓ Processor: Core i5-7300HQ
- ✓ RAM: 8GB

05 SIMULATION ENVIRONMENT:

The major parameters of the simulation environment are summarized in table 05. We shall investigate the protocol simulation.

Initially, a network of 20 clients adopts the simulation. The nodes are arranged in distinct topologies for mobility and non-mobility, once in a straight line and once at random on a 350x350 m grid. The speed of node movement varies by 1m/s. The model's movement is determined by its maximum travel speed and halt time. To generate a modest simulation, set the pause time to 1 second. We will simulate many scenarios. Each simulation will take one hour to complete.

The default values for some environment parameters are expressed below:

PARAMETERS	VALUE
Surface	350x350 (m)
Mote startup delay	1.000 (ms)
Time simulation	1 hour
Random-seed(random speed)	123,456
Speed of nodes	1s
Break time	4s
Number of servers	[5-6]
Number of client	[20,50]

Table 05: simulation environment.

In other hand, we must state the mote properties we utilized in each environment simulated in all-cases. T-mote s Sky, the following characteristics were highlighted in the table06:

Platform	Tmote Sky
MCU	8MHz TI MSP430F1611
Wireless transceiver	CC 2420 2.4 GHz, IEEE 802.15.4 radio
RAM	10K
ROM	48K flash ROM
Communication range	50m(in-doors)/200m(out-doors)
Operating system	Contiki-OS

Table 06: Characteristics of the Tmote Sky sensor node platform.

06 SIMULATION SCENARIO:

We built a simple subject to evaluate protocol, which is the movement of inmates inside the prison using CoAP to interact with captive devices and mobility to move mote across the area. The organization of prisoners in all prisons around the world is currently losing control of moving inmates around within it, so we attempted to study the most common movements in order to create a common simulation while keeping track of them by automating the doors in hallways to be opened and closed. Keeping track uses a distributed optical system in which the prisoner stands by the hallway's doors and after a set amount of time has passed, the doors automatically open for the next hallway, providing safety for guards and saving time and forces.

The smart prison includes a communications device, a controller, surveillance cameras, and a network of nodes, some of which are clients and others are servers. Mounted on clothing, servers are installed in gates, and connection between the two creates our system.

When the client-node moves, the prisoner will stand near a door in the range of the server-node, and the server-node received a request to open a specific door using CoAP, so that each client-node of converging columns sends data to its nearest server-node, and the server-node, in turn, communicates with GETAWAY. In this situation, COAP performance is evaluated using criteria (power consumption, throughput, and delay).

07 CoAP SIMULATION USING COOJA:

In this section, we will run the protocol CoAP in COOJA and connect the Border router to the server. The steps to follow are as follows:

- ✓ **First:** We open a new simulation, Open motes menu >> add motes >> create new motes type>>sky files are necessary to run CoAP applications. In order to create the motes: border-router.c, er-example-server.c, er-example-client.c.

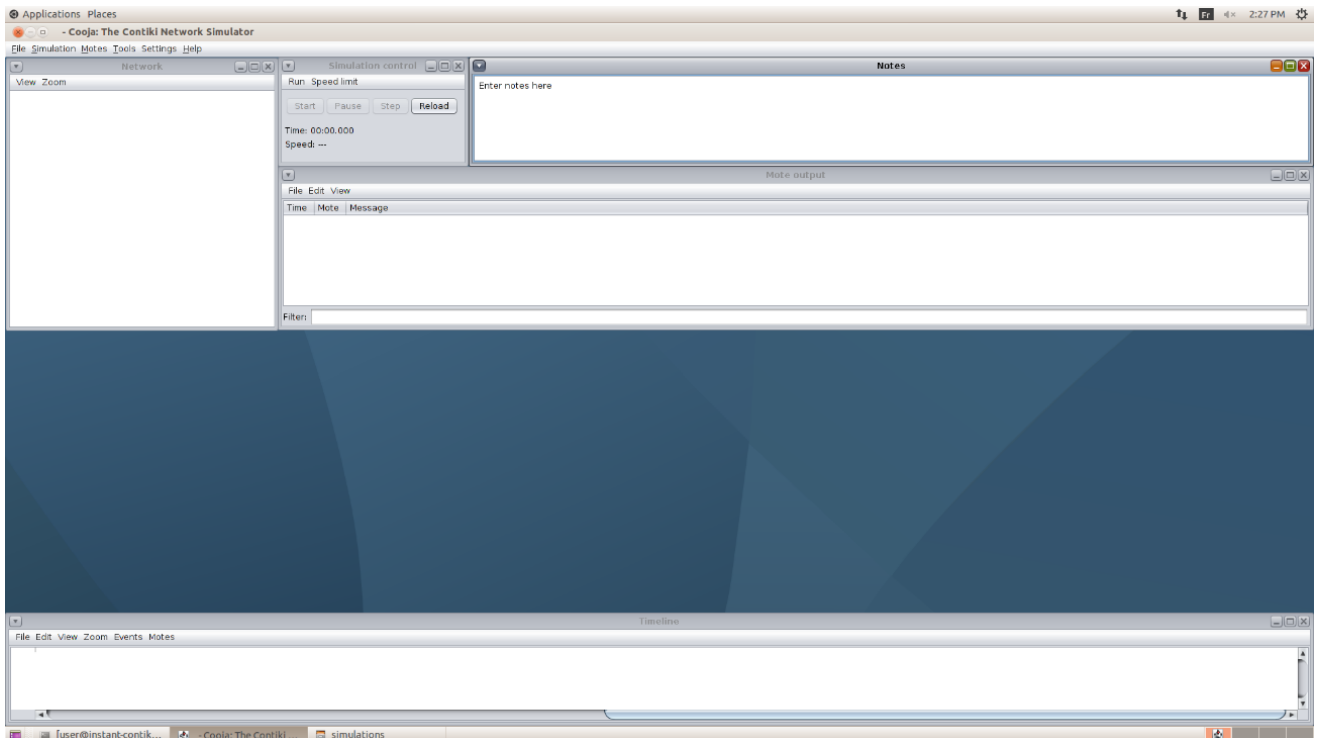


Figure26: COOJA window

- ✓ **Second:** To create border router motes: home/user/contiki/examples/ipv6/rpl-border-router/border-router.c, Choose the file in location >> compile >> create >> Add motes.

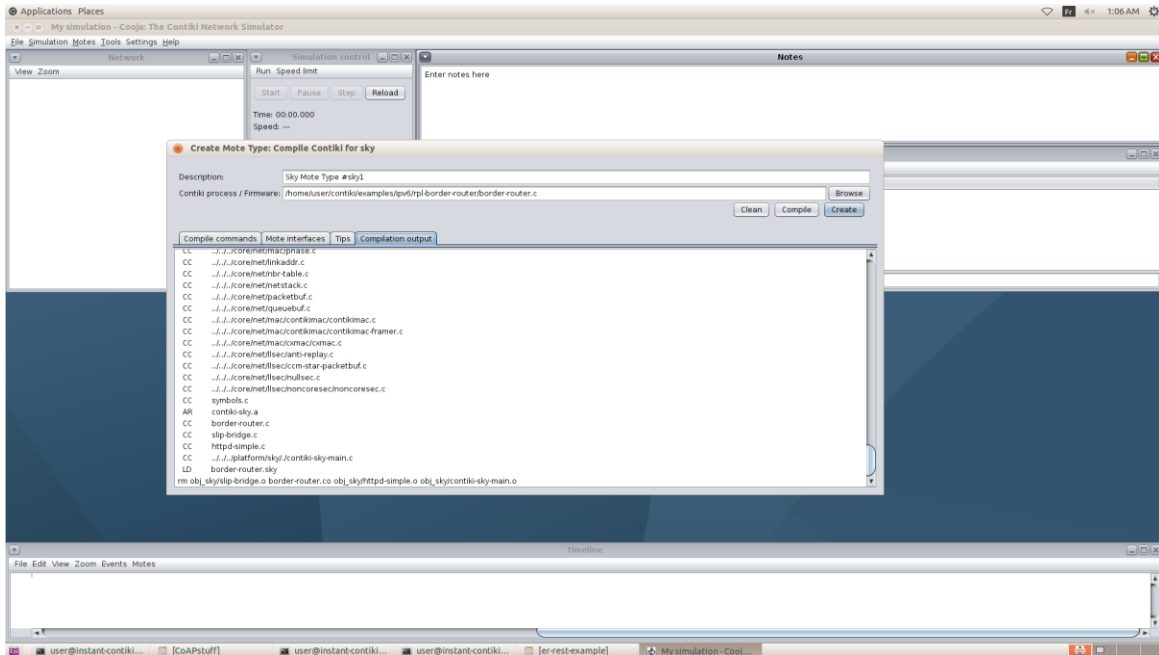


Figure27: Border Router Mote

- ✓ **Third:** To create server motes : /home/user/contiki/examples/er-rest-example/er-example-server.c, choose a file in location >> compile >> create >> choose server >> Add motes.

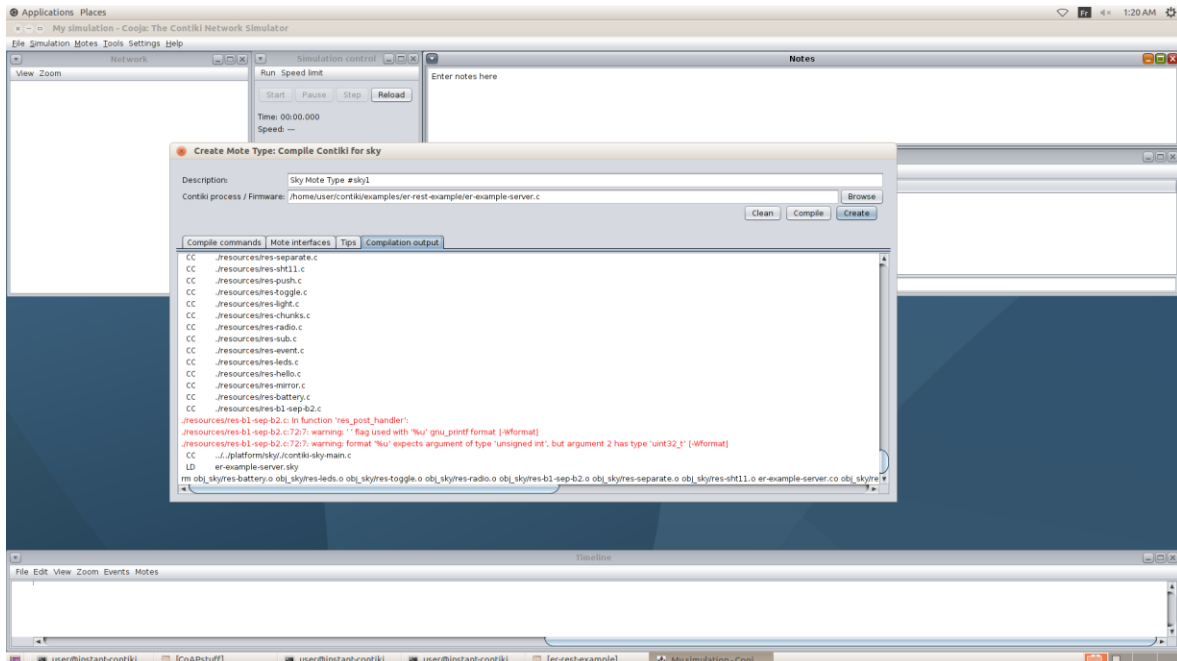


Figure28: CoAP server mote

- ✓ Fourth: To create client motes : /home/user/contiki/examples/er-rest-example/er-example-client.c, choose a file in location >> compile >> create >> choose server >> Add motes.

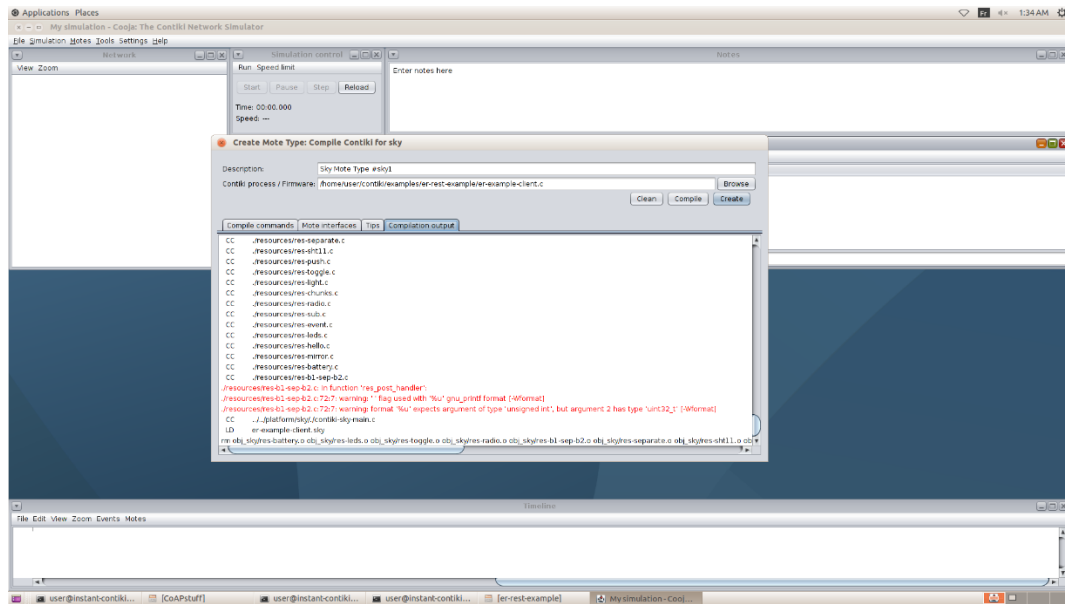


Figure29: CoAP client mote

- ✓ Fifth: And in order to make a connection between the border router and other nodes, we must enable a bridge by Right Click Border Router Node -> Mote tools for Sky 1 -> Serial Socket (SERVER) -> start.

On the other hand, we open a new terminal, and in the following path /home/contiki/examples/ipv6/rpl-border-router/ we do the command: make connect-router-COOJA.

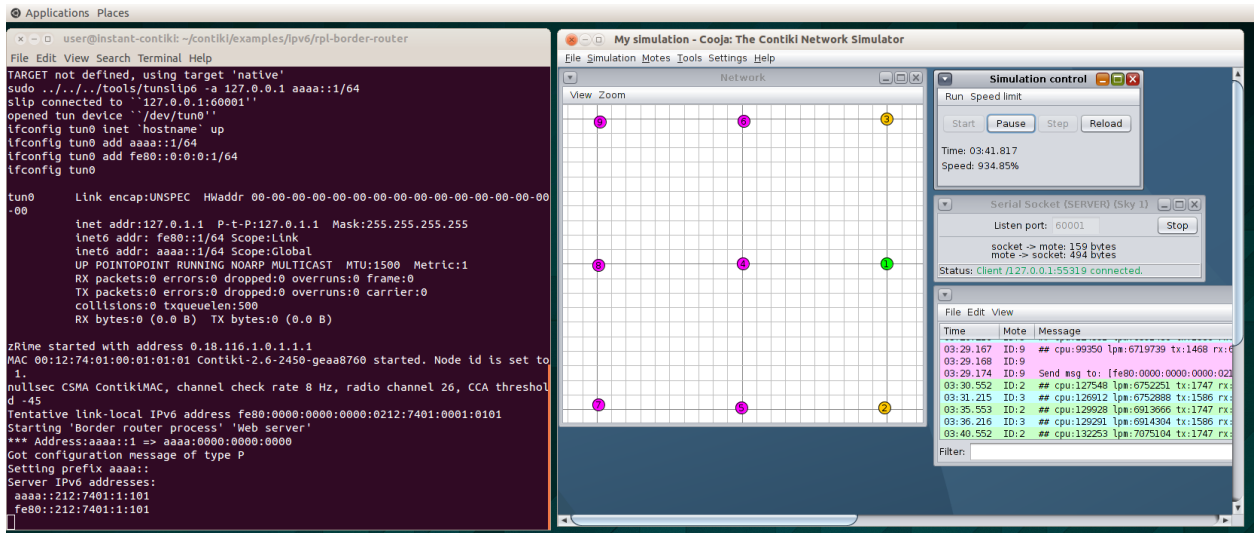


Figure30: border router and CoAP server

- ✓ Sixth: sensors can be reading by using IPV6 addresses, we used Libcoap library to test the communication by dropping the Intend server address with the Uri-CoAP,

To test it out u have to access to the file: libcoap/examples/, then by using this command: `coap-client -m post coap://[aaaa::212:7402:2:202]/actuators/toggle 1`

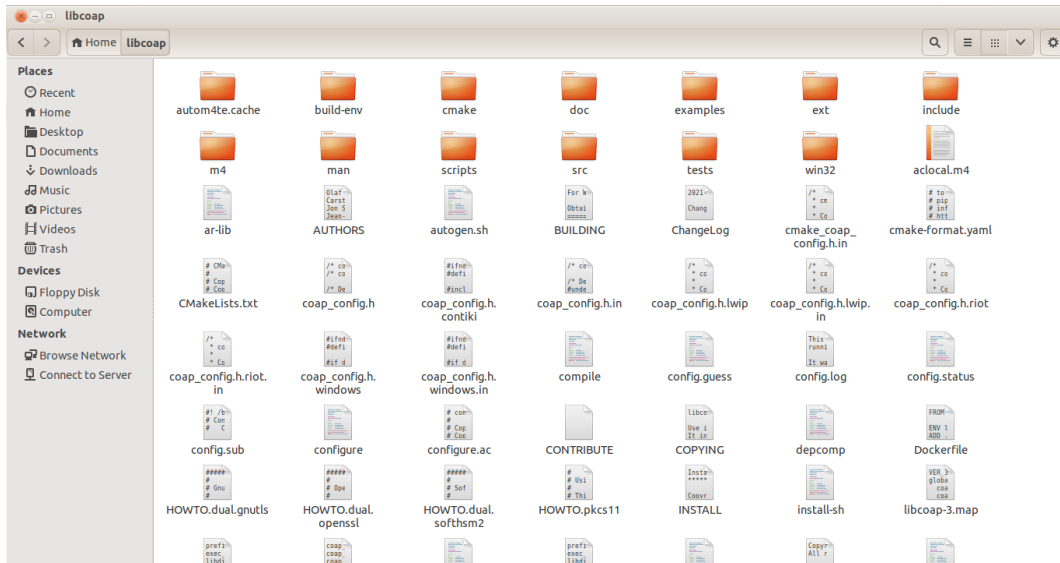


Figure31: Libcoap file.

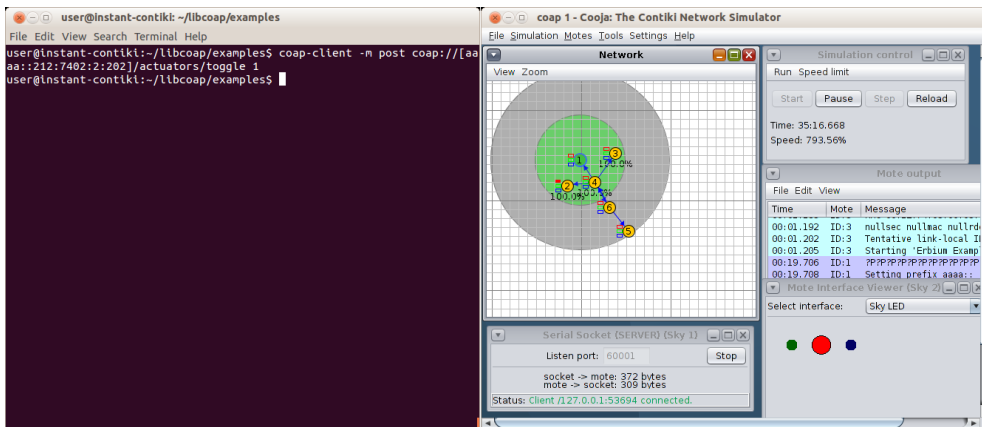


Figure32: Communication CoAP Server.

The above images show the libcoap plugin to open the ipv6 address and read the sensor values. In the first picture, the toggle value for Red LED is sent from the client-coap.c file by selecting the POST request, Upon receiving the Red LED is glowing in the mote that indicate that the server-node in accepting the inputs remotely.

08 MOBILITY OF NODE IN COOJA:[24][25]

This section lead us to enabling the mobility in Cooja Simulator. This feature can be used to simulate and test mobile ad-hoc network protocols in the Cooja Simulator. After following all the steps mentioned we will be able to successfully simulate mobile nodes in our Cooja simulator.

8.1 Downloading and Displacing Mobility file:

After loading file of mobility, we added the file to certain extension: Contiki/tools/cooja/apps/.

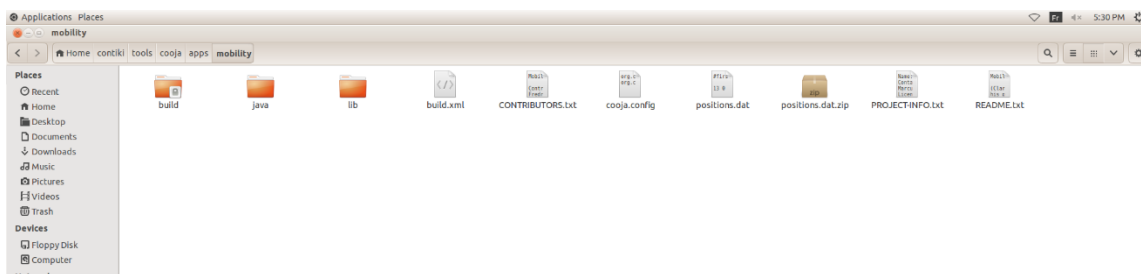


Figure33: displacement of mobility file

8.2 Modifications and Changes on Mobility file:

Since the version of Contiki we are using contiki3.0 explained why the mobility doesn't support this one version, so it should be there so updates in this file.

Starting with the extension that included in build.xml file:



Figure34: Change extension in build.xml file

The next step is to change the path of cooja inside cooja.config file:

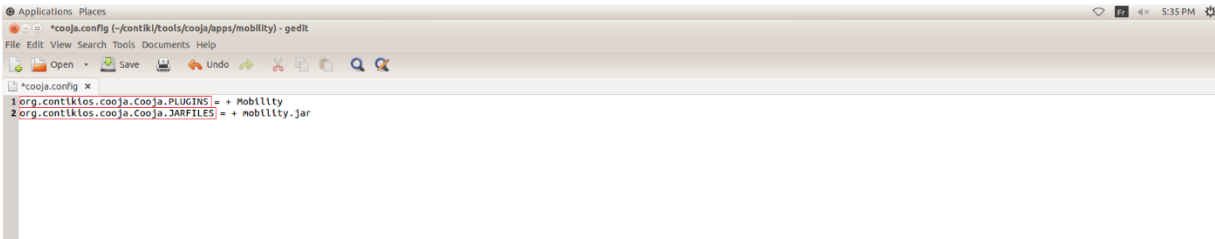


Figure35: changing cooja path in cooja.config

After doing all this changes we need to change cooja path inside Mobility.java file:

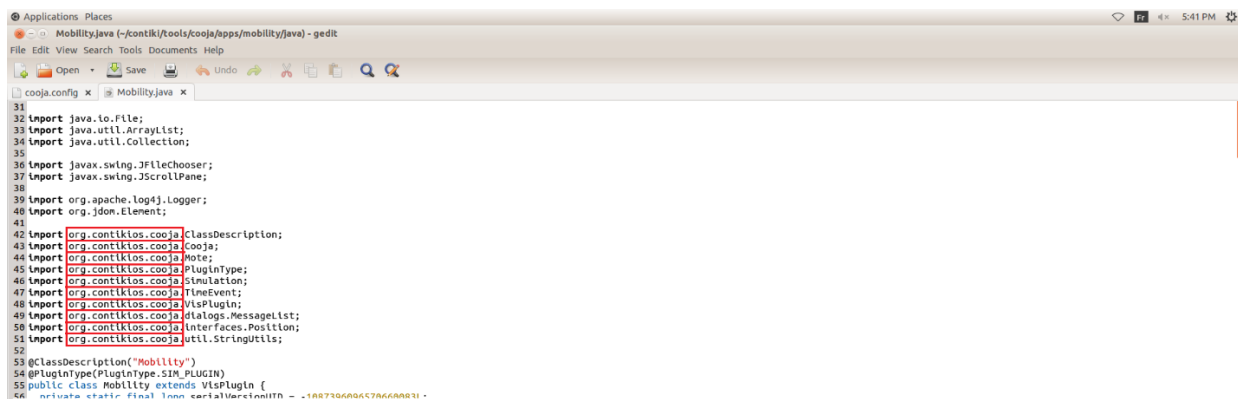


Figure36: changing cooja path in Mobility.java file.

Then we opened a terminal and access to the next extension:

Contiki/tools/cooja/apps/mobility/, and we run the next command: sudo ant jar.

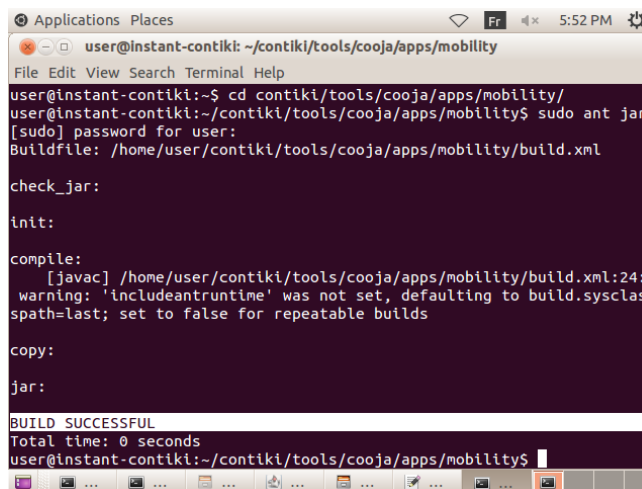


Figure37: activating mobility file in cooja.

8.3 Enabling the Mobility Plugin in Cooja:

While we are done with all past steps, we can finally enable mobility tool in cooja:

Starting with running cooja: `cd contiki/tools/cooja. sudo ant run.`

In Cooja, Settings > External Tools Path...

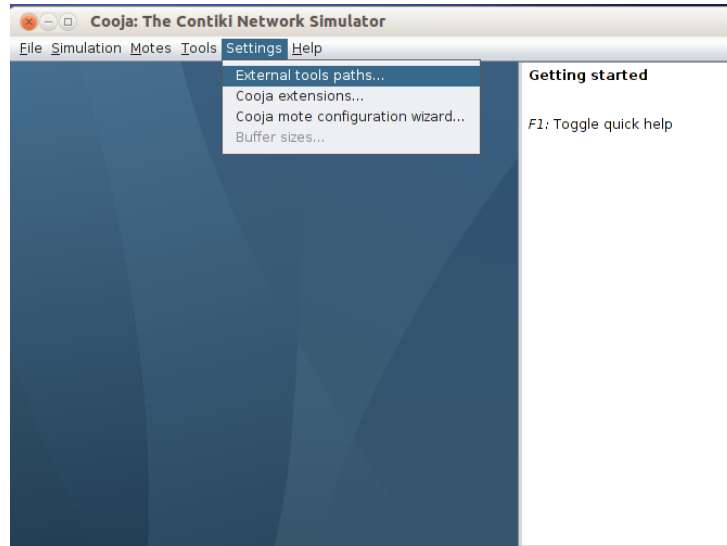


Figure38: Adding mobility to cooja extensions.

Edit Settings Screen will pop-up:

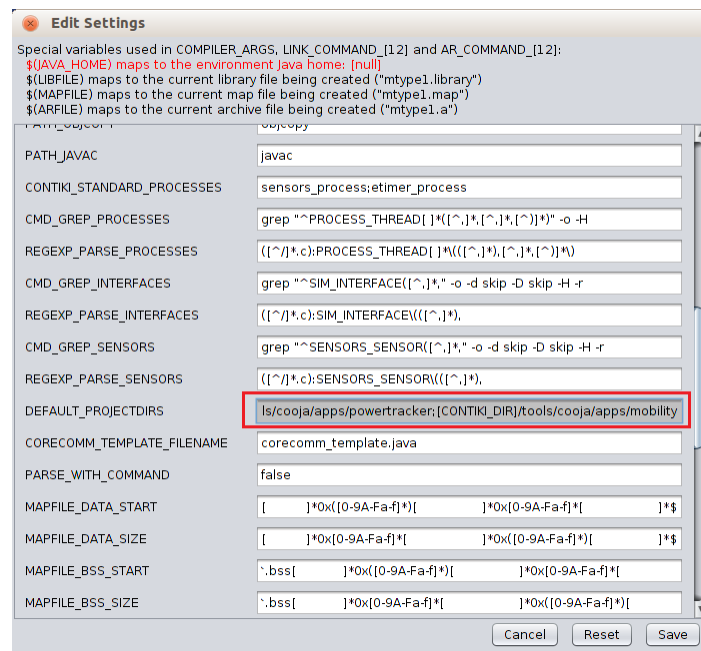


Figure39: Adding mobility path to cooja simulator.

To activate the mobility in cooja, after running it go to:

Settings>Cooja Extensions Cooja Extensions Window will pop-up. Scroll down to mobility select it and click on “Apply for the session”

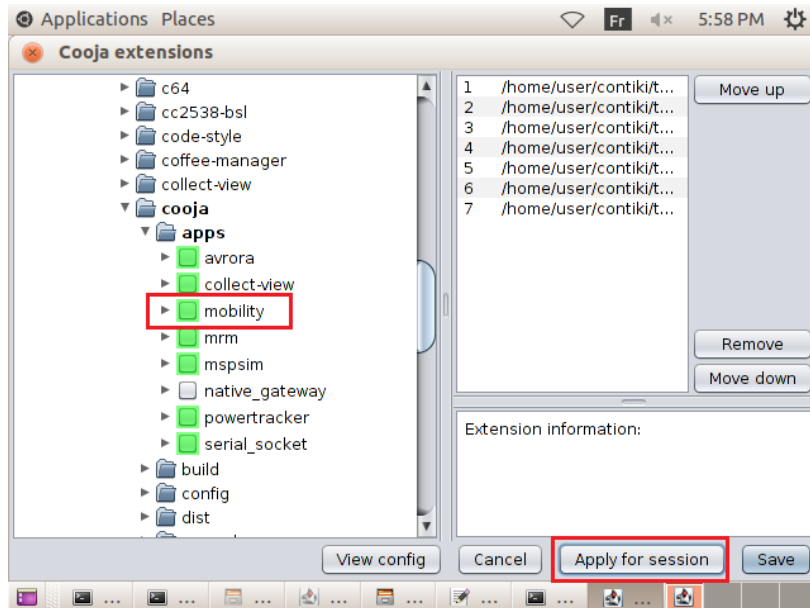


Figure40: Enabling mobility tool.

Now we can see that the mobility to enabled to use:

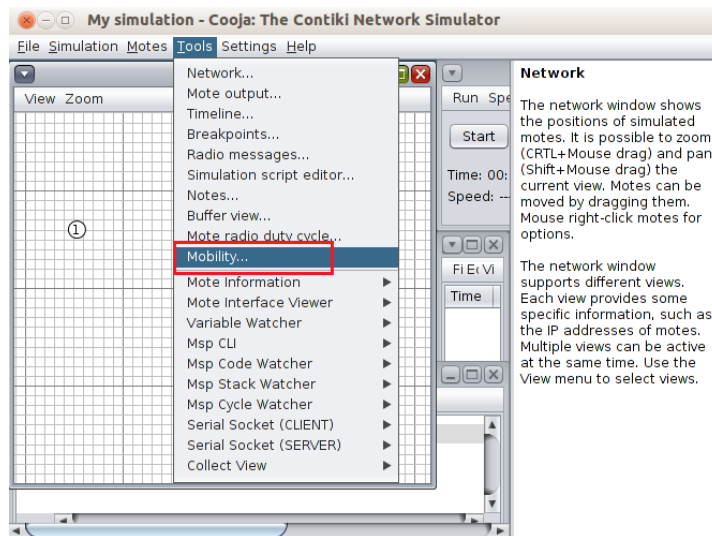


Figure41: mobility tool

09 PERFORMANCES EVALUATION EXPERIMENTS:

In order to evaluate the mobility with CoAP protocol performances, we use in our study two parameters (nodes number, mobility of nodes) where we study each parameter with power consumption, delay, and throughput. We did this by simulating the protocol and modifying the files provided in the system by adding some codes.

In each time we will change the number of nodes, or the way of nodes movements, after it we take the results and transform them in curves in order to analyze the protocol.

9.1 Experiment 1: Fixed nodes.

In this experiment we will study the throughput, delay, energy consumed and packet loss to the number of servers and clients, The first is simulated by 20, 50 clients, we have tried these conditions in different topologies (square, linear).

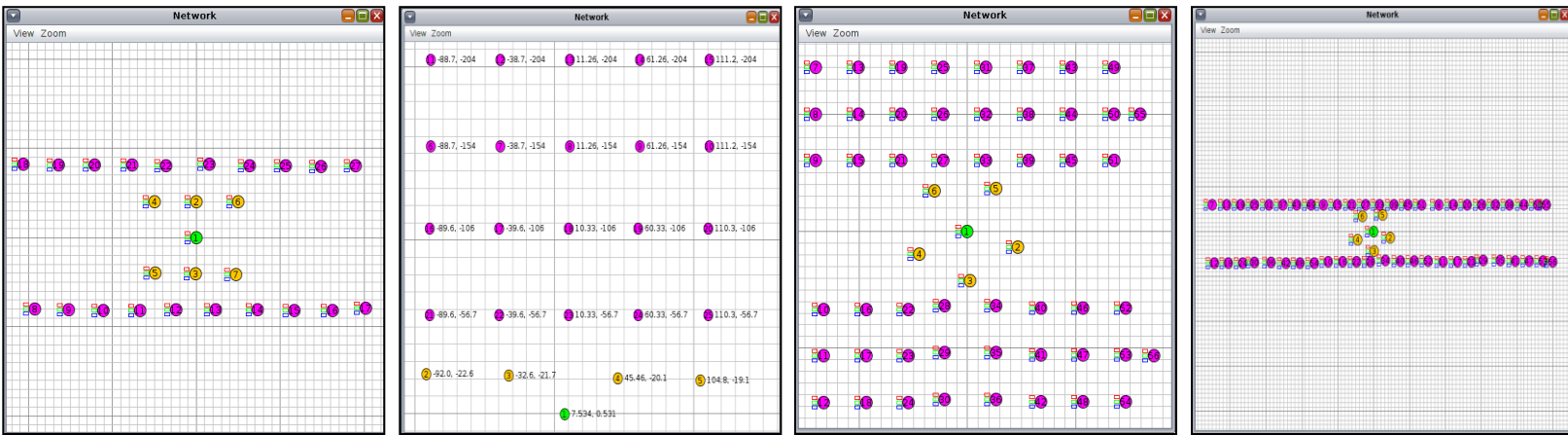


Figure42: Simulation For 20, 50 clients (linear, square).

9.2 Experiment 2: Mobility of nodes

In otherwise this experiment will be focusing on study the throughput, delay, energy consumed and packet loss to the number of servers and clients, The first is simulated by 20, 50 clients, we have tried these conditions in different topologies (square, liner).

In order to obtain the necessary measurements (delay, throughput, power consumption, packet loss). Here are some adjustments we've made to the Contiki OS files. We were also able to configure the COOJA simulator thanks to these modifications

in the Contiki system, allowing us to accomplish our work efficiently. We'll take a look at a few of these files:

- ✓ The er-core-engine.c: this file is responsible on the treatment of packages in coap protocol with form of protocol engineered.

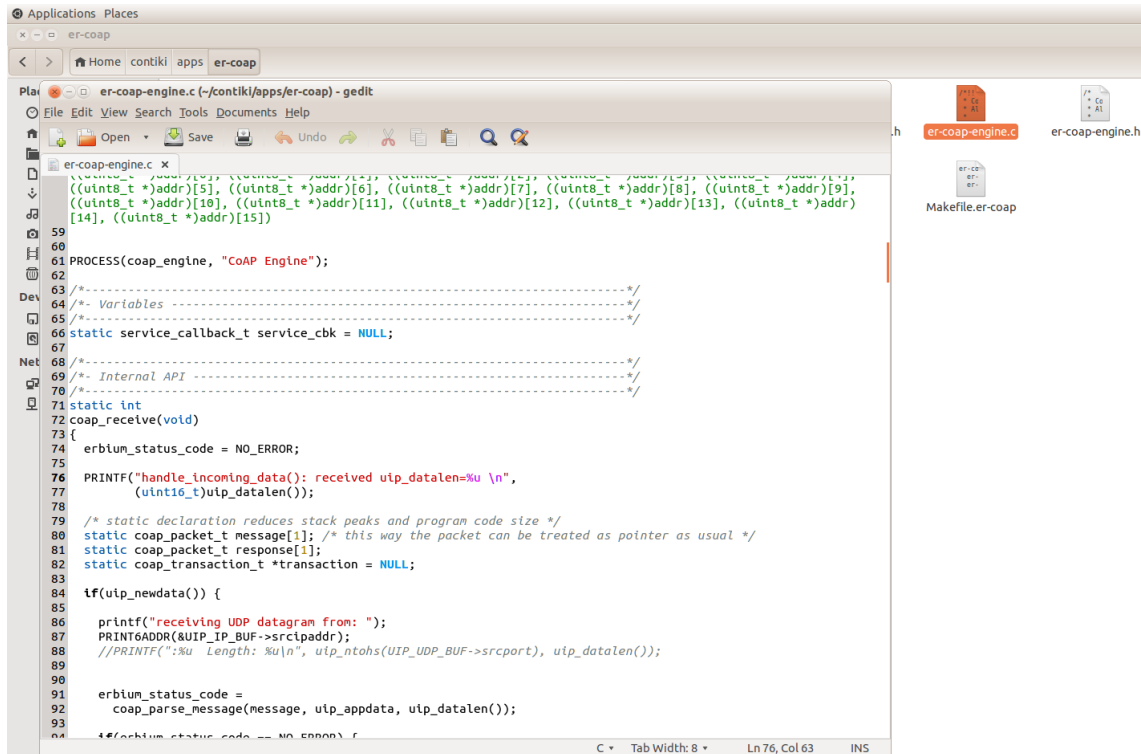


Figure43: Add Modifications To er-coop-engine.c file

- ✓ The powertrace.c: this one file is responsible on tracking the energy parameters in our simulation by printing out the values.

10 CONCLUSION:

We have given a brief presentation of the steps of simulation in this chapter for the purpose of simulation of the CoAP protocol. We started by defining the validation methods and some fields areas in which they are used, and then we moved on to present the programs used, as well as the simulation environment. Finally, we provide various simulations by increasing the time in each one (number of nodes, mobility of nodes).



CHAPTER 04

EVALUATION OF PERFORMANCES

Chapter 04: PERFORMANCES EVALUATION.

01 INTRODUCTION:

The selection of a model, the evaluation using a simulation approach, and the interpretation of the measurements acquired are all part of the simulation evaluation of a system's performance. For the study of architectures and protocols in diverse network contexts, a significant variety of simulation models have been built (number of nodes, mobility, etc.). They've been commonly employed in routing protocol evaluations.

In this chapter, we will review the results acquired from the simulations in the previous section, and we will conclude the effectiveness of the CoAP protocol in terms of the Internet of Things, both with and without the use of mobility.

02 CRETERIAS OF EVALUATION:

2.1 Energy Consumption:

The energy consumption of a network is the sum of all node's used energy, where a node's used energy is the sum of all energy needed for communication, include the transmitting (pt), receiving (pr), and idling (pi). The overall energy consumption is equal to the total number of packets sent in the network, assuming each transmission consumes an energy unit.

Power consumption= (Transmit/19.5 mA + Listen /21.8 mA +CPU power/1.8 mA +LPM/0.0545 mA)/ (32768).

2.2 Packet loss:

The total number of packets sented from the source versus The total number of packets losted while delivering them.

Packet loss= $((N_send - N_receive) * 100) / N_send$

03 EVALUATION PERFORMANCE METHODE:

With the modifications we added in the Contiki OS, each client and each server is ready to give the values we use to get the final results and appear through a "Mote output " window in COOJA.

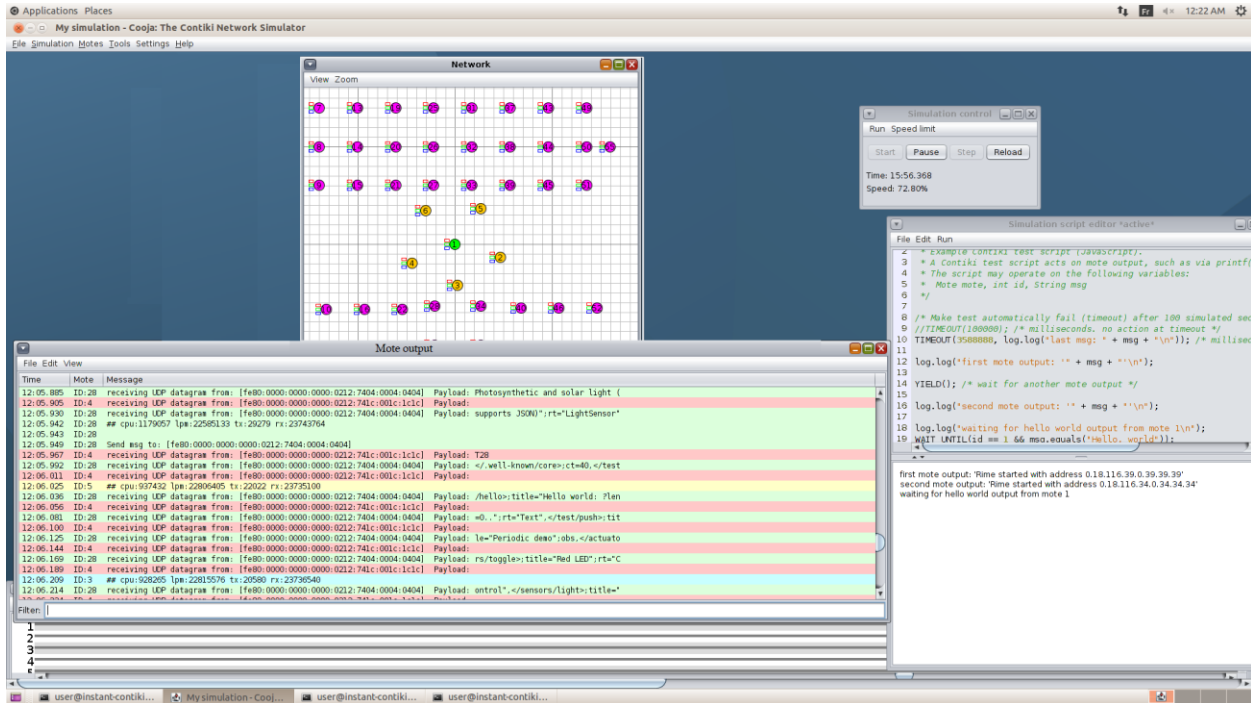


Figure46: The final results are in Mote output.

The purpose of using “Mote output” is to get the values of packet loss, and energy consumption, we will get values after time ended which is 59 minutes.

And every time we change the paraments to obtain different results in order to compare them. each time, we change the number of clients, servers and topology, and also by using mobility over motes.

3.1 Results simulation experiment 1: Fixed Nodes.

The following table presents the results extracted from the simulation of the first experiment which is based on the parameter: the number of clients and servers. We calculated the measures of the characteristics (packet loss, energy consumption) in each time (20,50 clients /6,7 servers), By keeping the same simulation surface:

Number Of Clients (Topology)	Number Of Servers	Packet Loss	Energy consumption
20 (Linear)	4	11%	851550 J
20 (Square)	4	14%	830000 J
50 (Linear)	6	7.78%	757000 J
50 (Square)	6	16%	737855 J

Table 07: Criteria values depended on Clients and Servers number
 From the results of the above table, the following Graphic shapes were constructed:

Figure 47 shows the percentage of packet loss related to number of Clients and Topology

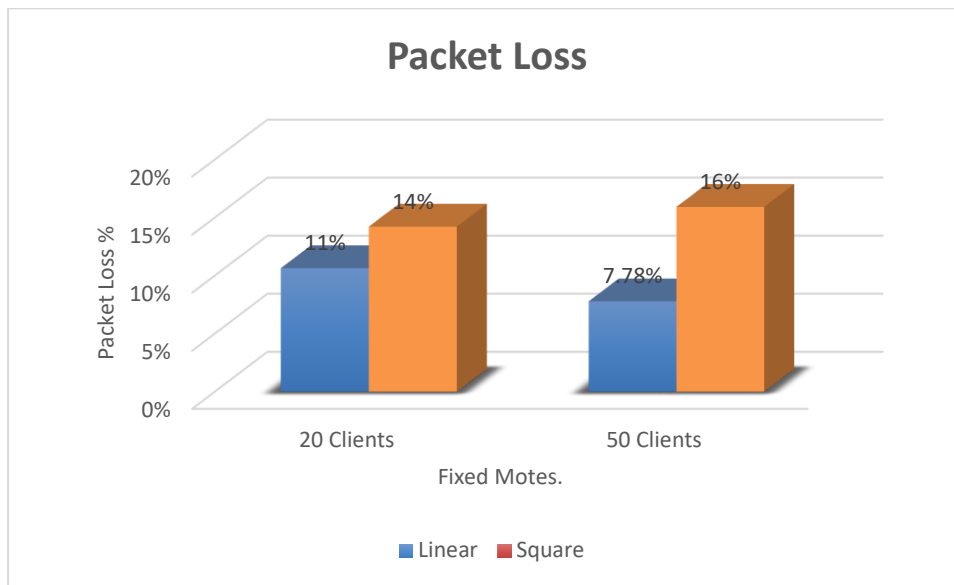


Figure 47: Curves of Packet loss against the Number of COAP Clients and Topology.

We notice in figure (47) when 20 Clients took the percentage of packet loss in either topology is acceptable, then when increasing the number of Clients to 50 we notice that is small different in either topology in the packet loss.

Figure 48 shows the energy consumption values in relation to the number of Clients and topologies:

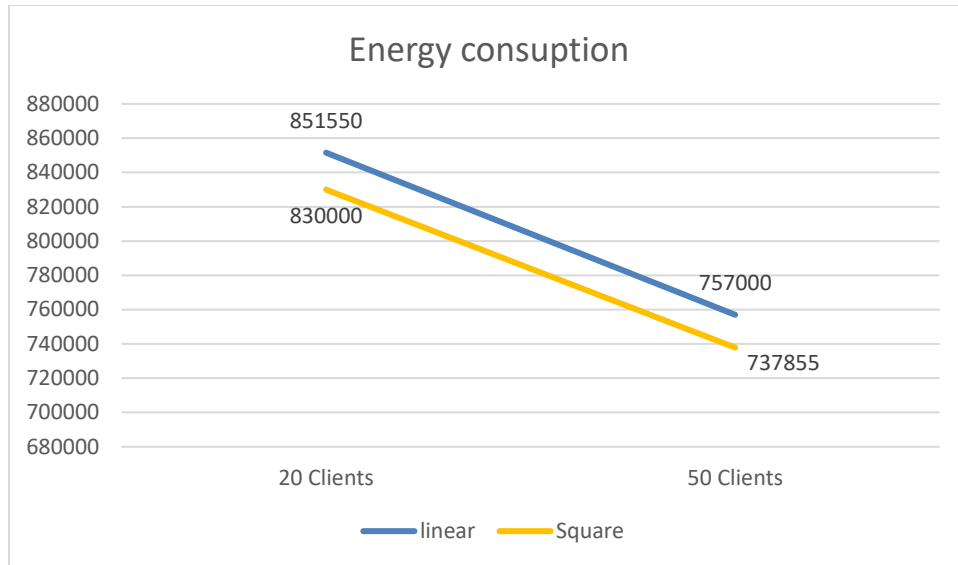


Figure 48: energy consumption values in relation to the number of Clients and topologies

In figure 48, we notice that the values of energy consumption when we used 20 Clients with a linear topology less consuming then using the square topology and we can say the same about the 50 Clients.

Analyze

While studying packet loss, and energy consumption note with a gradual increase in the number of clients and with changing topology's type:

- A number of Clients increases, the radio traffic increases due the difficulty of accessing the Border Router explained by the presence of percentage of packet loss.
- The effectiveness of the topology's type on the percentage of packet loss explained the results.
- Since the number of motes depended on consuming energy, that's explain the values of energy consumption.

3.2 Results Simulation Experiment 2: Mobility nodes.

The following table present the results extracted from the simulation of the second experiment in which we relied on mobility nodes calculated the characteristic (packet loss, energy consumption) each time, by keeping same number of Clients node and changing in number of servers.

Number Of Clients (Topology)	Number Of Servers	Packet Loss	Energy consumption
20 (Linear)	5	12%	840000 J
20 (Square)	5	21%	836500 J
50 (Linear)	7	18%	831550 J
50 (Square)	7	22%	810000 J

Table 08: Criteria values depended on mobility of Clients and Servers number
From the results of the above table, the following curves were constructed:

Figure 49: shows the packet loss related the nodes mobility:

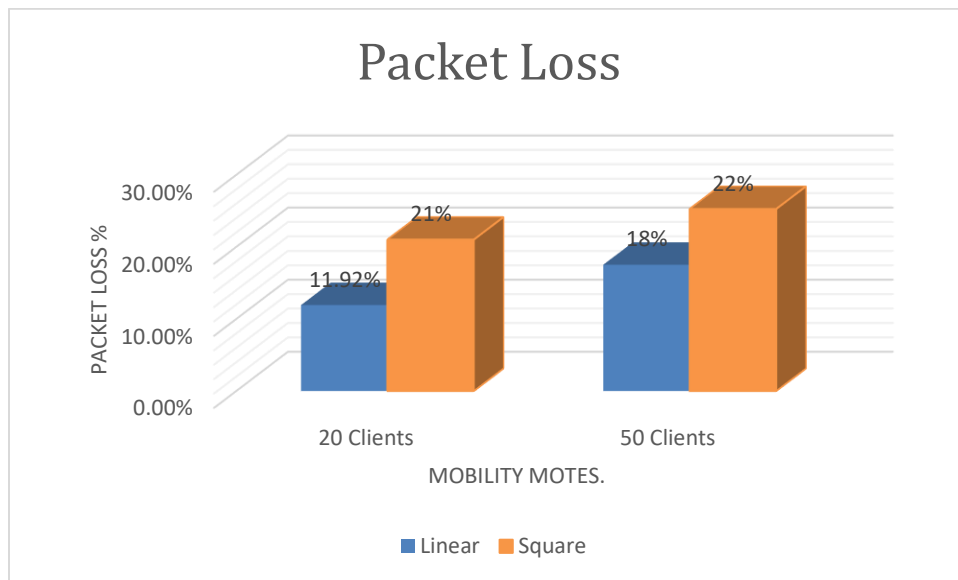


Figure 49: curves of packet loss against the number and mobility of motes.

We notice in figure 49 when we added the mobility to nodes that effected more on the increasing results of packet loss upper to the value: 22%.

Figure 50: shows the energy consumption values in relation with mobility nodes:

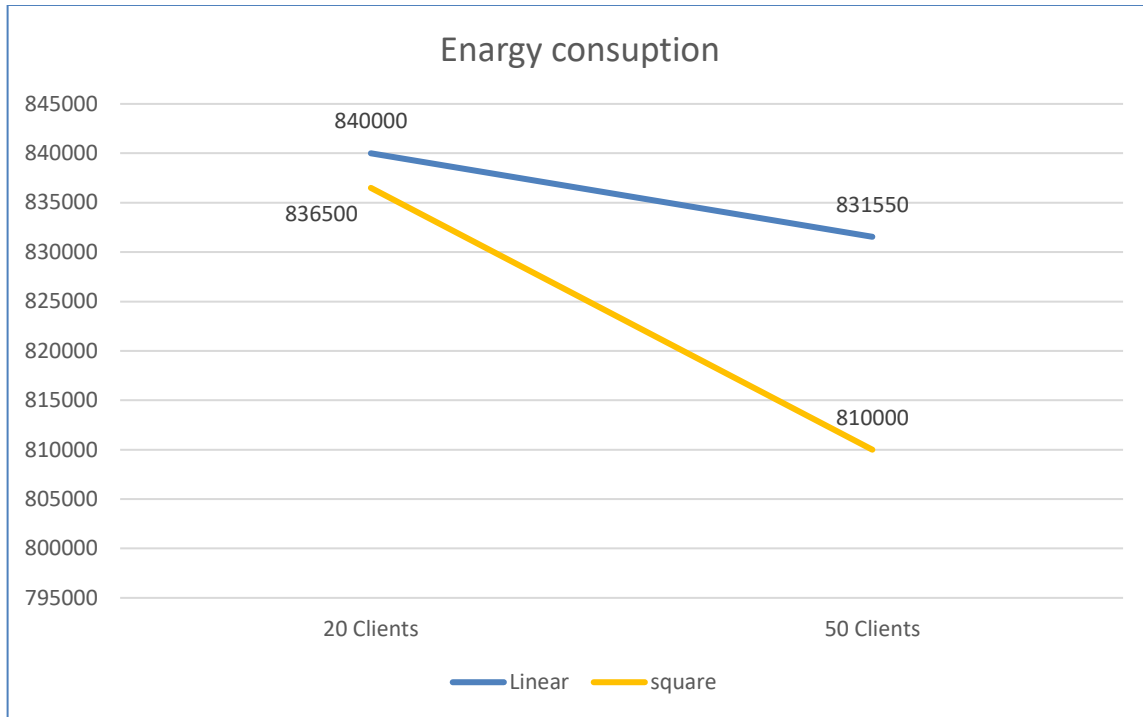


Figure 50: energy consumption values in relation with mobility nodes.

In figure 50s, we notice that the values of energy consumption when we used 20 Clients with a linear topology less consuming then using the square topology and we can say the same about the 50 Clients.

Analyze

While studying packet loss, and energy consumption note with gradual increase in the number of clients and with changing topology's type all that after adding the mobility to nodes:

- A number of Clients increases, and the movability of nodes, the radio traffic increases due the difficulty of accessing to the Border Router explained by the presence of percentage of packet loss.
- The effectiveness of the topology's type on the percentage of packet loss explained the results in the increasing the it values.
- Since the number of motes depended on consuming energy, that's explain the values of energy consumption.
- Adding the mobility to motes kept the values of energy consumption almost the same because the mobility is not related to be consuming energy.

04 EVALUATION RESULTS:

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



Not affected




Affected

	Number of Clients	Number of servers	Topologies	Mobility
Packet Loss				
Energy consumption				

Table 09: compare standards by ratio for different transmissions

Through the table 09: we have reached:

- The packet loss gets effected when the number Clients is huge.
- The packet loss gets effected with type of topology.
- The packet loss gets effected with odes mobility
- The energy consumption gets effected with both number of clients, servers and topology, but it doesn't get effected by the mobility.











	Packet Loss	Mobility
Scalability		
Reliability		
Energy consumption		
Network Topology		

Table 10: CoAP and IoT Aspect Results.

Through table 10, we have reached:

- Reliability is a burden on the application layer protocol, CoAP in this case, since it bypasses unreliable UDP at the transport layer. The results demonstrate the reliability of the protocol in terms of successful packet reception. Since the retransmission factor allows for increased reliability, the incorporation of the retransmission process of the CoAP protocol guarantees a significant increase in its reliability.
- The CoAP design over UDP has a great impact on power consumption, this is measured through different simulation scenarios which prove the continuous increase in power consumption.
- The CoAP protocol is not particularly sensitive to increasing the number of Clients and servers, despite the low rate of message delivery and without creating a normal operating condition (such as stopping the request delay). This ensures the scalability of the bound devices.

04 Conclusion:

In this chapter, we presented an illustrative study to evaluate the performance of the CoAP protocol, we also focused on the results of every experiment by comparing different condition.

What we had focused on:

- Studying and Evaluating CoAP protocol without the mobility.
- Studying and Evaluating CoAP protocol with using mobility (Mobile Motes).
- Extract the results from each study.
- Comparison of the performance in each experiment.

After all what we notice, that the effectiveness of the mobility on CoAP protocol specially on the packet loss, as we can see in results of undelivered packets that it gets increased in our environment network topologies, even thou the energy consumption did Not get effected in our study by the movements of motes.

With all this results we can finally say that the CoAP protocol needed more then just upgraded, in the way it works and try to fixed the critical conditions that it would be in the future.



General Conclusion

General Conclusion:

The Internet of things is one of the biggest changes in innovation today, and it is very important that its products must be reliable and efficient, and respect all the required characteristics. In this work, we focused on the protocols part due to the multiplicity and difficulty of choosing the optimal protocol for the user, and specifically, we paid attention to the mobile nodes in the client's side, otherwise we also focused on evaluating the performance of the CoAP protocol by studying its performance in the mobility nodes side through simulations in the COOJA network simulator, complex conditions to see how effective this protocol.

Among the programs and tools, we used in this research is the COOJA emulator based on the Contiki-os operating system and the Libcoap as a client side to improve the communication between nodes in our simulator. In order to evaluate the protocol through some criteria: energy consumption, delay, and packet loss in order to find out how effective they are with characteristics Internet of Things This work was carried out in four phases:

- phase 1: a general study on the field of the Internet of things, its advantages, its use, characteristics, and protocols....so on
- phase 2: a theoretical understanding of CoAP and a theoretical study of it with the characteristics of the Internet of things
- phase 3: How to work with the COOJA emulator and mobility tool with some changes in Contiki-os file's code.
- phase 4: Efficacy and outcome study and protocol evaluation

We encountered many hurdles in our work, but the previous theoretical description of the protocol was rich for us.

Finally, regarding the scenario that we have relied on in our work, it was concluded that CoAP achieves reliability and scalability, and one of its weaknesses is its high energy consumption.

CoAP achieves high loss of packets, and is a good protocol for connecting to IoT applications.

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

- Evaluate the performance of CoAP with the characteristics that we did not use
- Validation by mathematical analysis method formal
- Improvement in the CoAP protocol so that it becomes the basic protocol in the Internet ofthings application layer(energy consumption) This is done by reducing the size of CoAP protocol's algorithms for the reduce processing time or adding a sleep feature to the node while the movement is up and when it is not active. At the end of this note, we have added several skills in evaluating the performance of the protocol by simulation method.

Bibliography Documents:

- [1]:Seung-Man Chun, Hyun-Su Kim and Jong-Tae Park"CoAP-Based Mobility Management for the Internet of Things".In Proceedings School of Electronics Engineering, College of IT Engineering, Kyungpook National University, pp.16061-16082
- [2] F P Mahimai Don Bosco, E Chitra and S.R. Ebenezer "A Survey of Low-Latency IoT System Using FPGA Accelerator" J. Phys.: Conf. Ser. 1964 062014.
- [3] Shelby. Z, Hartke. K, Bormann. C. (June 2014) Internet Engineering Task Force (IETF), Standards Track, <https://tools.ietf.org/html/rfc7252>.
- [4] C.Perera, A.Zaslavsky, P.Christen and D.Georgakopoulos ,“Sensing as a service model for smart cities supported by Internet of Things” , Trans.Emerging ,© 2013 , Ltd. DOI: 10.1002/ett
- [5]. OpenIoT Consortium. Open source solution for the internet of things into the cloud, January 2012. Available from: <http://www.openiot.eu> [08 April 2012].
- [6]. Commonwealth Scientific and Industrial Research Organisation (CSIRO). Phenonet: Distributed sensor network for phenomics supported by high resolution plant phenomics centre, csiro ict centre, and csiro sensor and sensor networks tcp., Australia, 2011. Available from: <http://phenonet.com> [20 April 2012].
- [7]. C. Perera, Y.Qin, Julio C. Estrella”Fog Computing for Sustainable Smart Cities: A Survey”,ACM Computing Surveys, Vol. 0, No. 0, Article 00, Publication date: 2017.
- [8]. Bandyopadhyay and A. Bhattacharyya, “Lightweight internet protocols for web enablement of sensors using constrained gateway devices,” in Computing, Networking and Communications (ICNC), 2013 International Conference on. IEEE, 2013, pp. 334–340.
- [9]. D. Thangavel, X. Ma, A. Valera, H.-X. Tan, and C. K.-Y. Tan. “Performance Evaluation of MQTT and CoAP via a Common Middleware,” in Intelligent Sensors, Sensor Networks and Information Processing, 2014 IEEE Ninth International Conference on. IEEE, 2014, pp. 1–6.
- [10]. B. Mishra, A. Kertesz “The use of MQTT in M2M and IOT system: A survey” [Online] Available: <https://ieeexplore.ieee.org/document/9247996/authors#authors>
- [11]. N. S. Han, “Semantic service provisioning for 6LoWPAN: powering internet of things applications on web,” Ph.D. dissertation, Institut National des Tel-ecomunications, 2015.

- [12]. N. Naik, P. Jenkins, P. Davies, and D. Newell, "Native web communication protocols and their effects on the performance of web services and systems," in 16th IEEE International Conference on Computer and Information Technology (CIT). IEEE, 2016, pp. 219–225.
- [13]. I. Grigorik, "Making the web faster with HTTP 2.0," Communications of the ACM, vol. 56, no. 12, pp. 42–49, 2013.
- [14]. A. Foster, "Messaging technologies for the industrial internet and the internet of things whitepaper," PrismTech.
- [15]: AlabbasAlhajAli. "Constrained Application Protocol (CoAP) For TheIoT ". Delivred May 2018
- [16]: LudmillaLucioSilva . "Internet Of Things Pros And Cons Of CoAP Protocol Solution For Small Devices ". Delivred 15/02/2015
- [17] Z. Shelby, K. Hartke and C. Bormann. "Constrained Application Protocol for Internet Of Things "Delivred 5/05/2014
- [18]Amélie Gyrard, "Concevoir Des Applications Internet Des Objets Sémantiques Interdomaine", T H È S E, Telecom Paristech, Ecole De L'institut Télécom - Membre De Paristech, 24 Avril 2015.
- [19]:Isam Ishaq, Jeroen Hoebeke 1, Floris Van den Abeele, Jen Rossey Ingrid Moerman and Piet Demeester "Flexible Unicast-Based Group Communication for CoAP-Enabled Devices",Department of Information Technology (INTEC), Ghent University - iMinds, GastonCrommenlaan 8 Bus 201, Ghent 9050, Belgium.
- [20] L. B. Saad, C. Chauvenet, And B. Tourancheau, "Simulation Of The Rpl Routing Protocol For Ipv6 Sensor Networks: Two Cases Studies," In Proc. Of The 2011 International Conference On Sensor Technologies And Applications (Sensorcomm'11), Nice, France. Iaria, September 2011.
- [23]: "Introduction A La Programmation En Langage Python" , Université Paris-Sud ,Méthodologie Licence Mpi S2 - Année 2015-2016
- [26]: AlessandroLudoviciAnd AnnaCalveras . "A Proxy Design To Leverage The Interconnection Of CoAPwireless Sensor Networks With Web Applications ".Dilvred 09/01/2015.
- [28]: Ludmilla Lucio .Silva,CompusHamosamd . "Internet Of Things – Pros And Cons Of CoAP Protocol Solution For Small Devices". UnivaerstelSbacken .02-15.2016

Web Bibliography:

[21]:<https://github.com/obgm/libcoap> 15:30 09/06/2022

[22] https://www.rocq.inria.fr/secret/Anne.Canteaut/COURS_C/cours.pdf Programmation En Langage C 11:06 22/06/2022

[24]:<https://ambar-prajapati.medium.com/cooja-mobility-plugin-6d43bb756751>
11 :1122/06/2022

[25]: <https://contikiemj.blogspot.com/2018/03/mobility-plugin-in-contiki-30-cooja.html> 11 :10
22/06/2022

[27]"Introduction A La Programmation En Langage Python", Université Paris-Sud ,
Méthodologie Licence Mpi S2 - Année 2015-2016

[29]:www.java.org ; 21:57,02/06/2022.

[30]Dr Omar Elloumi,Président De La Plénière Technique Onem2m,Url.16/06/2022