Ministère de l'enseignement supérieur et de la recherche scientifique

Université de Kasdi Merbah Ouargla

Faculté des nouvelles technologies de l'information et de communication

Département des mathématiques et de l'informatique

MEMOIRE DE FIN D'ETUDES

Pour l'obtention du diplôme de Master en informatique

Domaine : Mathématique et Informatique
Filière : Informatique
Spécialité: Administration et sécurité des Réseaux

Thème :

Etude comparative de protocoles de communication dans l'IOT

Réalisé par :

BENCHOULA Rayane

HANACHI Nour Elhouda

Membres du jury :

Encadreur : Mme.KHELILI Farida Khalida

Examinateur : M.BEKKARI Fouad Ben Abdellah

Examinateur : Mme.KORAICHI Wassila

Promotion : 2019/2020

Remerciement

C’est pour nous un plaisir autant qu’un devoir de remercier toutes les personnes qui ont pu contribuer de près ou de loin à l’établissement de ce projet, qui nous ont aidé, soutenu et ont fait sorte que ce travail ait eu lieu.

Ainsi, nous exprimons notre gratitude et nous tenons à remercier Mme KHELILI Farida qui nous a encadré, et qui n’a épargné aucun effort pour nous orienter afin qu’on puisse mener à bien ce projet.

Nous tenons aussi à remercier les membres du jury pour leur précieux temps accordé à l’étude de notre mémoire.

Enfin, nous remercions affectueusement nos parents, qui ont toujours su nous faire confiance et nous soutenir sans compter dans nos études.

Dédicace

Je dédie ce mémoire

À mes chers parents (Sabrina et Mustapha) et à ma grand-mère (Naima)

qui m'ont toujours soutenu tout au long de

ma vie et de mon parcours scolaire merci pour l'éducation

et les conseils que vous m'avez offerts et qui me permettent toujours

d'avancer dans la vie

A mon grand-père M'Hamed

A mes chèresfrères et sœur Chihab,Dacil et Sophie

A mes chères tantes Kenza, Sara et fatma

A mes petits cousins et cousines

A tous mes ami(e)s avec lesquels j'ai partagé des moments

de joies et de bonheur

A mon binôme Selma

BENCHOULA Rayane

Dédicace

Je dédie ce mémoire

A mes très chers parents pour leur soutient et encouragement durant toutes mes années d'études et sans lesquels je n'aurais jamais réussi.

A mes chères sœurs Widad, Rahma, Yousra et Lyna ainsi qu'à mon frère Yanis.

J'espère toujours être à la hauteur de vos espérances.

A mon binôme Rayane,

A mes chers amis : Meriem, Maroua, Iman, Naima, Oussama, et Marouane,

A toute personne ayant contribué à ce travail de près ou de loin.

A tous mes professeurs et enseignants que j'ai eu durant tout mon cursus scolaire et qui m'ont permis de réussir dans mes études.

HANACHI Nour Elhouda

Résumé

L'internet of things (IoT) est un vaste réseau qui comprend différents dispositif intelligent. IoT permet à des objets de collecter et d'échanger des données et de communiqué entre eux.

Afin d'assurer cette communication, l'IoT fait appel à des différents protocoles de communication qui vont permettre à ses objets de communiquer, parmi ses protocoles il y'a le protocole CoAP et le protocole MQTT.

Ces protocoles sont toujours en développement afin de permettre une meilleure interaction, et chaque protocole a ses propres caractéristiques et certaines performances.

Pour étudier leurs fonctionnements et les performances qu'ils offrent, il existe plusieurs méthodes parmi elles la simulation grâce à des différents outils comme le simulateur COOJA.

Mots Clés : Internet of things, protocoles de communication, CoAP, MQTT, comparaison, Simulation, COOJA.

Abstract

The Internet of Things (IoT) is a vast network that includes different smart devices, it allows items to collect and exchange data.

The IoT uses protocols that allow objects to communicate among its protocols there are the CoAP protocol and the MQTT protocol.

These protocols are still in development to allow better interaction, and each protocol has its own characteristics and performance.

In order to know their operation and the performance that they offer there are several methods among them the simulation thanks to simulators like the cooja simulator.

Keywords: Internet of things, communication protocols, CoAP, MQTT, comparison, Simulation, COOJA.

Table des matières

Liste des figures

Liste des tableaux

Liste des abbreviations

AMQP : Advanced **M**essage **Q**ueuing **P**rotocol

CoAP: Constrained **A**pplication **P**rotocol

DTLS : Datagram **T**ransport **L**ayer

DDS : Data **D**istribution **S**ervice

HTTP: Hyper**t**ext **T**ransfer **P**rotocol

IEEE: Institute of **E**lectrical and **E**lectronics **E**ngineers

IHM: Interfaces **H**ome-**m**achine

IOT: Internet **o**f **T**hings

IPv6 : Internet **P**rotocol **v**ersion **6**

ISO: International **O**rganization for **S**tandardization

MQTT: Message **Q**ueuing **T**elemetry **T**ransport

M2M: Machine **toM**achine

NAT: Network **A**dresse **T**ranslation

QoS : Quality **o**f **S**ervice

TCP: Transmission **C**ontrol **P**rotocol

UDP : User **D**atagram **P**rotocol

WSN : Wireless **s**ensor **n**etwork

XMPP : Extensible **M**essaging and **P**resence **P**rotocol

Introduction générale

L'internet et le réseau informatique ont connus un énorme développement ce qui a fait naître l'internet des objets (connu sous le nom de IoT: internet of things) et grâce à ce développement presque tout est devenu connecté, maintenant il n'y a pas que les ordinateur et les smart phones qui ont accès à internet et permettent à l'humain d'accéder en ligne de plus en plus d'appareille ont accès à internet comme les objets du quotidien l'électroménager par exemple.

Ces objets contiennent des dispositifs informatiques reliés par les réseaux qui vont permettre à un objet d'échanger des données.

L'interaction entre les objets nécessite différents type de technologies telles que les protocoles de communication ou ce qu'on appelle les protocoles de communication de l'IoT.

Jusqu'à présent Il n'existe pas de standardisation pour les protocoles IoT et les utilisateurs ne savent pas quel protocole choisir car chaque protocole a ses propres caractéristiques et performances c'est pour cela que nous avons décidé d'étudier deux des protocoles de l iot les plus connus qui sont le protocole MQTT qui a été standardisé par OASIS et le protocole CoAP qui a été standardisé par l'IETF.

La simulation d'un protocole permet de suivre et de voir comment il fonctionne, afin de comprendre et d'étudier les protocoles MQTT et CoAP nous avons simulé ces deux derniers avec le simulateur COOJA pour pouvoir connaître et analyser les performances de chaque protocole pour pouvoir faire une comparaison entre eux.

Notre mémoire est constitué d'une introduction générale, quatre chapitres qui sont :

Chapitre 1 : Nous présentons dans ce 1er chapitre le concept de l'IoT, son fonctionnement ainsi que ses différentes caractéristiques.

Chapitre 2 : Ce chapitre sera consacré à la communication dans l'IoT ou nous présentons les différents protocoles de l'IoT et leurs caractéristiques.

Chapitre 3 : Nous présentons dans ce chapitre les deux protocoles MQTT et CoAP et nous allons expliquer leur fonctionnement et nous ferons une comparaison.

Chapitre 4 : Ce dernier chapitre sera consacré a la simulation de nos deux protocole MQTT et CoAP.

Et enfin nous terminerons avec une conclusion générale.

Chapitre I : Concept de l'Internet des objets

I. Chapitre 1 Concept de l'IOT

I.1 Introduction :

L'internet des objets (l'IoT) représente l'échange des objets connecté à internet, ces objets peuvent émettre et recevoir des informations. Grâce à l'impact positif de l'IoT sur notre quotidien notre vie quotidienne sera simplifiée et notre bien-être sera amélioré.

Dans ce chapitre on va présenter l'IoT il y'aura une définition, quelque domaines d'application que l'IoT offre, le fonctionnement et l'architecture de l'IoT et aussi les protocoles que l'IoT utilise et enfin on donnera une définition de l'objet connecté.

I.2 L'Internet des objets

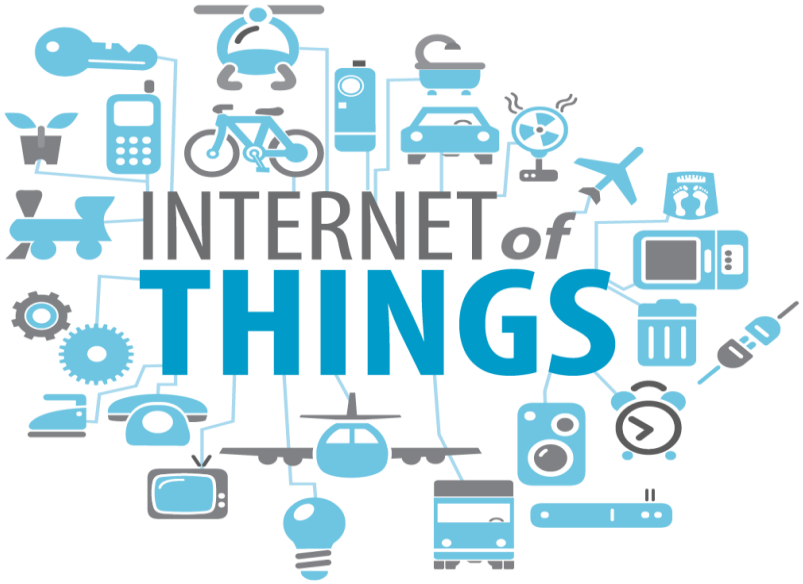


Figure 1.1 L'internet des objets

I.2.1 Définition

L'internet des Objets désigne une technologie d'avant-garde, où les objets traditionnellement non connectés qui nous entourent (comme des lampes, machines, vêtements, etc.), qu'ils soient physiques ou virtuels, ont désormais la capacité de communiquer entre eux en temps réel. Ce réseau d'objets là permet le partage de leurs données par l'intermédiaire d'une plateforme Cloud et ce, sans intervention humaine. Grâce à l'optimisation des interactions entre les humains et les machines et à la multiplication des flux de données, que les objets connectés offrent la possibilité de définir les besoins précis d'un individu, de sorte à lui offrir un bien ou un service unique. [1]

Dans l'IOT, tout Objet est potentiellement connecté à Internet est capable de communiquer avec d'autres Objets. Ceci engendre de nouveaux risques liés notamment à la confidentialité, l'authenticité et l'intégrité des données échangées entre les Objets [2].

L'IEEE[1] définit l'IoT comme un « réseau d'éléments chacun muni de capteurs qui sont connectés à Internet » [8]

Il n'existe pas de définition standard, unifiée et partagée de l'internet des objets Certaines définitions insistent sur les aspects techniques de l'IoT, tandis que d'autres se concentrent plutôt sur les usages et les fonctionnalités [9].

I.2.2 Objet connecté

I.2.2.1 Définition

Les objets connectés interagissent avec leur environnement par le biais de capteurs (température, vitesse, humidité, vibration…). Dans l'Internet des Objets, un objet peut aussi bien être un véhicule, une machine industrielle ou bien une place de parking [14].

I.2.2.2 Caractéristiques d'un objet connecté

Généralement, un objet connecté est caractérisé par [13]:

Identité : pour que les objets soient gérables il est essentiel que chaque objet connecté possède une identité unique qu'il lui propre et qui le distingue des autres objets du système.
Interactivité : Un objet n'a pas besoin d'être connecté à un réseau à tout moment.
Programmable: l'objet connecté doit être programmé et piloté à distance via un ordinateur, une tablette ou un Smartphone.
Sensibilité : un objet a la capacité de percevoir son environnement et peut collecter ou transmettre des informations à celui-ci.
Autonomie : cette caractéristique est, peut-être, la caractéristique la plus importante pour l'objet connecté. On désigne par cette caractéristique la capacité de l'objet d'agir sans l'intervention d'un tiers.

[1] Institute of Electrical and Electronics Engineers

I.2.3 Domaines d'application de l'Internet des objets

L'utilisation de l'internet des objets permettra le développement de plusieurs applications dans différent domaine, Plusieurs domaines d'application sont touchés par l'IOT, Parmi ces domaines nous citrons: le domaine du transport, le domaine de l'industrie, le domaine de La santé, domaine d'agriculture et le domaine de la domotique.

- **Domaine du transport :** Grâce à l'IoT il existe des voitures connectées et intelligentes et autonomes. L'utilisation de ces voitures permet de sauver des vies, ces voitures peuvent avoir un impact positif sur l'environnement car elles permettent de réduire le trafic routier.
- **Domaine de l'industrie :** La technologie IoT permettra un suivi total des produits, de la chaîne de production, jusqu'à la chaîne logistique et de distribution en supervisant les conditions d'approvisionnement. Cette traçabilité de bout en bout permet aux usines d'améliorer l'efficacité de ses opérations, d'optimiser la production et d'améliorer la sécurité des employés. [2]
- **Domaine de La santé :** Dans le domaine de la santé, l'IoT permettra le déploiement de réseaux personnels pour le contrôle et le suivi des signes cliniques, notamment pour des personnes âgées, les objets connectés permettent de suivre la tension, le rythme cardiaque, la qualité de respiration ou encore la masse graisseuse. Ceci permettra ainsi de faciliter la télésurveillance des patients à domiciles, et apporter des solutions pour l'autonomie des personnes à mobilité réduite. [2]
- **Domaine d'agriculture** : L'usage des objets connectés se démocratise dans l'agriculture. De nombreuses amélioration ou découlent la gestion des engins agricoles, la maîtrise de l'irrigation ou la gestion optimisée des intrants, que la surveillance de la croissance des plantes ou encore la prévention des risques météo. De quoi renouveler en profondeur les pratiques dès cette activité ancestrale, grâce à l'analyse des données récoltées et au pilotage de plus en plus fin des exploitations [3]
- **Domaine de la domotique :** La domotique, ou maison connectée, représente l'utilisation de l'Internet des Objets dans une maison. Grâce à la domotique, les tâches du quotidien deviennent plus simples et facile.

Les smart cities

Efficace et innovant le concept smart cities offre aux habitants une bonne qualité de vie avec une consommation de ressources minimale grâce à une combinaison intelligente des infrastructures (énergie, transport, communication) aux différents niveaux hiérarchiques (ville, quartier, bâtiment).

Exemple réel de smart city [46] :

La ville de Santander, en Espagne, a été l'une des premières, en Europe, à se lancer dans une stratégie smart city à grande échelle. Les capteurs de la ville produisent plus de 200 000 datas par jour, fluidifiant la vie quotidienne et offrant des économies d'énergie à la municipalité. Pourtant l'aspect collaboratif peine à s'imposer. Etat des lieux.

Santander s'est lancée dans une vaste politique d'installation de capteurs, des lampadaires aux poubelles, et de développement d'applications dédiées aux habitants, faisant de cette ville de 180 000 habitants un laboratoire grandeur nature des stratégies smart city appliquées aux métropoles intermédiaires de l'Europe. La municipalité a installé plus de 20 000 capteurs, générant plus de 200 000 données quotidiennes, collectées, traitées et agglomérées par la ville. Santander connaît ainsi, en temps réel et en tout point de la ville, le taux de CO_2 ou de NO_2, le niveau de bruit, l'intensité lumineuse.

Figure 1.2 Exemple de smart city [47]

I.2.4 Fonctionnement de l'IoT

I.2.4.1 Composants de l'IoT

Un système IoT réunit de nombreux acteurs et composants technologiques qui assure le bon fonctionnement d'un système IoT. Une solution d'IoT s'articule autour de 5 composants essentiels que sont [4]: Les objets (capteurs), Le réseau (connectivité), Les données, Les informations et Les applications d'exploitation.

- **Les objets :** pour capter des données de valeur. Ce sont les équipements actifs ou passifs pouvant générer de la donnée exploitable et créatrice de valeur pour les utilisateurs. Les objets sont composés d'éléments passifs : les capteurs, et pour certains d'éléments actifs les rendant capables de traitements d'enrichissement de la données et de transmission de celle-ci. Les données sont aussi diverses que les métiers. Nous pouvons aussi bien avoir des données de température, d'humidité, de positionnement, de temps de fonctionnement, de niveau, d'alerte…[4]
- **Le réseau :** pour transmettre les données. Les réseaux sont le maillon prépondérant d'un projet d'IoT, ils doivent répondre à un critère d'usage : La couverture de la zone d'usage des objets : Sur un campus, Sur une ville, à l'ensemble de la planète.[4]
- **Les données :** Dans un projet d'IoT, les données sont nos diamants bruts. Il s'agit surtout des éléments bruts que nous récoltons depuis les objets ou les outils de processus industriels pour l'IoT. Afin de créer de la valeur pour les utilisateurs de ces données, il est absolument nécessaire de les stocker, archiver et sauvegarder dans des bases de données et de correctement structurer cette dernière. En effet une base de données correctement structurée améliorera la performance des services IoT d'exploitation [4].
- **Les informations :** les informations sont les résultantes des données traitées, corrélées et analysées. Ces informations doivent elles-aussi être stockées, archivées et sauvegardées dans des bases de données [4].
- **Les applications d'exploitation :** Les applications d'exploitation sont en principe les interfaces Homme-machine (IHM) dans lesquelles nous pouvons visualiser les données sous forme de tableau de bord [4].

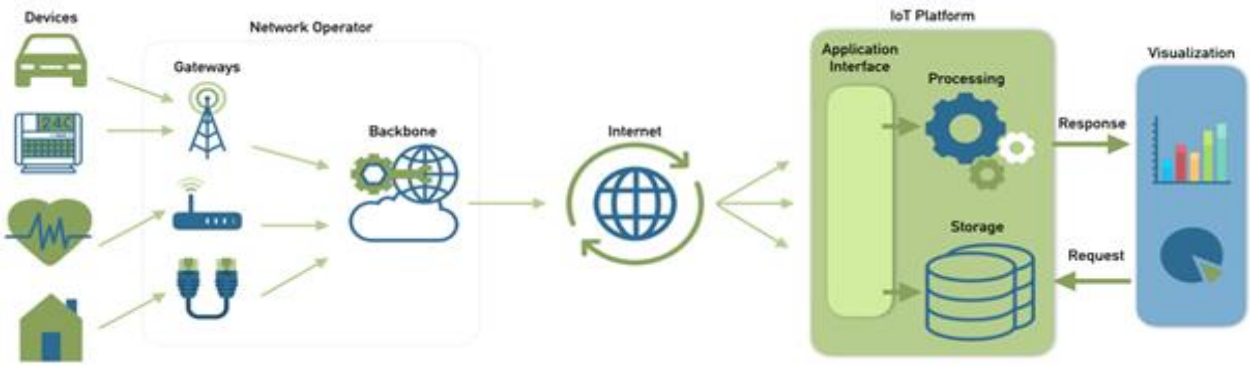


Figure 1.3 Fonctionnement de l'Internet des Objets [10]

I.2.4.2 Technologies de l'IoT

L'IoT permet l'interconnexion des différents Objets intelligents via l'Internet. Et pour le bon fonctionnement d'un système IoT, plusieurs systèmes technologiques sont nécessaires. Bien qu'il existe plusieurs technologies utilisées dans le fonctionnement de l'IoT, nous mettons l'accent seulement sur quelques-unes qui sont : M2M, RFID, WSN et le Bluetooth

- **M2M :** Avec le M2M (Machine to Machine), il s'agit tout simplement d'utiliser des réseaux télécom existants à savoir de la 2G jusqu'à la 4G. Majoritairement, il s'agit d'abonnements souscrits auprès des opérateurs de téléphonie pouvant donner accès uniquement à des volumes de données [4].
- **RFID :** c'est une technologie sans fil qui est utilisée pour l'identification des objets[5], elle englobe toutes les technologies qui utilisent des ondes radio pour identifier automatiquement des Objets ou des personnes. C'est une technologie qui permet de mémoriser et de récupérer des informations à distance grâce à une étiquette qui émet des ondes radio. Il s'agit d'une méthode utilisée pour transférer les données des étiquettes à des Objets, ou pour identifier ces Objets à distance. L'étiquette contient des informations stockées électroniquement peuvent être lues à distance [11].
- **WSN :** c'est un ensemble de nœuds qui communiquent sans fil et qui sont organisés en un réseau coopératif. Chaque nœud possède une capacité de traitement et peut contenir différents types de mémoires, un émetteur-récepteur RF et une source d'alimentation. Il peut aussi tenir compte des divers capteurs et actionneurs [7].
- **Le Bluetooth :** Il s'agit d'une technologie radio de moyenne portée (environ 10m) qui permet d'envoyer des messages de grande taille et en grande quantité. Cependant, cette connectivité ne se suffit pas à elle-même car elle nécessite une tierce technologie pour transférer et stocker les données. D'autre part, il s'agit d'un moyen de communication disposant d'un grand débit, puisqu'il repose sur la bande de fréquence 2,4 GHz tout comme le Wifi [4].

I.2.5 L'architecture de l'IoT

Une architecture technique est un cadre créé pour permettre aux concepteurs et aux développeurs de considérer le système dans son ensemble et de le décomposer en sections. Selon les normes mondiales 1 (GS1), «une architecture de référence est une fondation pour permettre l'intégration des diverses technologies dans les applications IoT» [34].

Il existe de nombreux proposition d'architectures pour l'IoT parmi les architectures qui ont été proposées, il y'a l'architecture a cinq couches proposée par Wu et al et a trois couche proposé par IEEE (voir figure 1.3).En ce moment y'a pas une architecture universellement acceptée.

A. L'architecture à trois couches: Cette architecture se compose de:

La couche application (application layer) est chargée de fournir des services spécifiques à l'application à l'utilisateur. Il définit diverses applications dans lesquelles l'Internet des objets peut être déployé, par exemple, les maisons intelligentes, les villes intelligentes et la santé intelligente.[35]
La couche réseau (abstract layer) est responsable de la connexion à d'autres objets intelligents, périphériques réseau et serveurs. Ses fonctionnalités sont également utilisées pour la transmission et le traitement des données des capteurs.[35]
La couche perception (things layer) est la couche physique, qui possède des capteurs pour détecter et recueillir des informations sur l'environnement. Il détecte certains paramètres physiques ou identifie d'autres objets intelligents dans l'environnement.[35]

B. L'architecture à cinq couches : Cette architecture a cinq couches se compose de :

La couches de perception et d'application leur rôle est le même que celui de l'architecture à trois couches.
La couche de transport (transport layer) transfère les données du capteur de la couche de perception à la couche de traitement et vice versa via des réseaux tels que sans fil, 3G, LAN, Bluetooth, RFID et NFC.[35]
La couche de traitement (processing layer) est également appelée couche middleware. Il stocke, analyse et traite d'énormes quantités de données provenant de la couche transport. Il peut gérer et fournir un ensemble diversifié de services aux couches inférieures. Il utilise de nombreuses technologies telles que les bases de données, le cloud computing et les modules de traitement des méga données.[35]
La couche métier (business layer) gère l'ensemble du système IoT, y compris les applications, les modèles commerciaux et de profit, et la confidentialité des utilisateurs.[35]

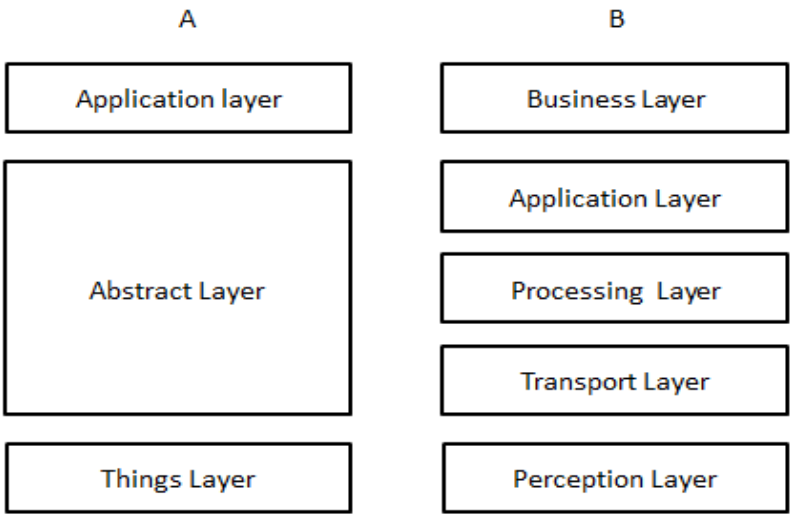


Figure 1.4 Architectures IoT: (A) Architecture à trois couches IEEE P 4213 [30]. (B) Architecture à cinq couches [35]

I.2.6 Les protocoles de l'IoT

De nombreuses normes IOT sont proposées pour faciliter et simplifier les tâches des programmeurs d'applications et des fournisseurs de services. Différents groupes ont été créée pour fournir des protocoles, y compris les efforts menés par le W3C, IETF, EPC global, IEEE et l'ETSI [37].

L'IoT ambitionne de faire communiquer chaque système avec tous autres au moyen de protocoles communs. La mise en application à une large échelle du concept d'IOT apparaît largement tributaire d'une standardisation de la communication entre objets dite M2M.

Au niveau de la couche de liaison : le standard IEEE 802.15.4 est plus adapté que l'Ethernet aux environnements industriels difficiles.
Au niveau réseau : le standard 6loWPan a réussi à adapter le protocole IPV6 aux communications sans fil entre nœud à très faible consommation.
Au niveau routage : l'IETF a publié en 2011 le standard RPL.
Au niveau de la couche application : le protocole CoAP qui tente d'adapter http, beaucoup trop gourmand aux contraintes des communications entre nœuds à faible consommation [38].

L'IoT a sa propre pile de protocoles, différente des autre pile de protocoles comme le modèle OSI et le protocole TCP / IP.

Nous allons voir les différents protocoles de l'architecture à cinq couches

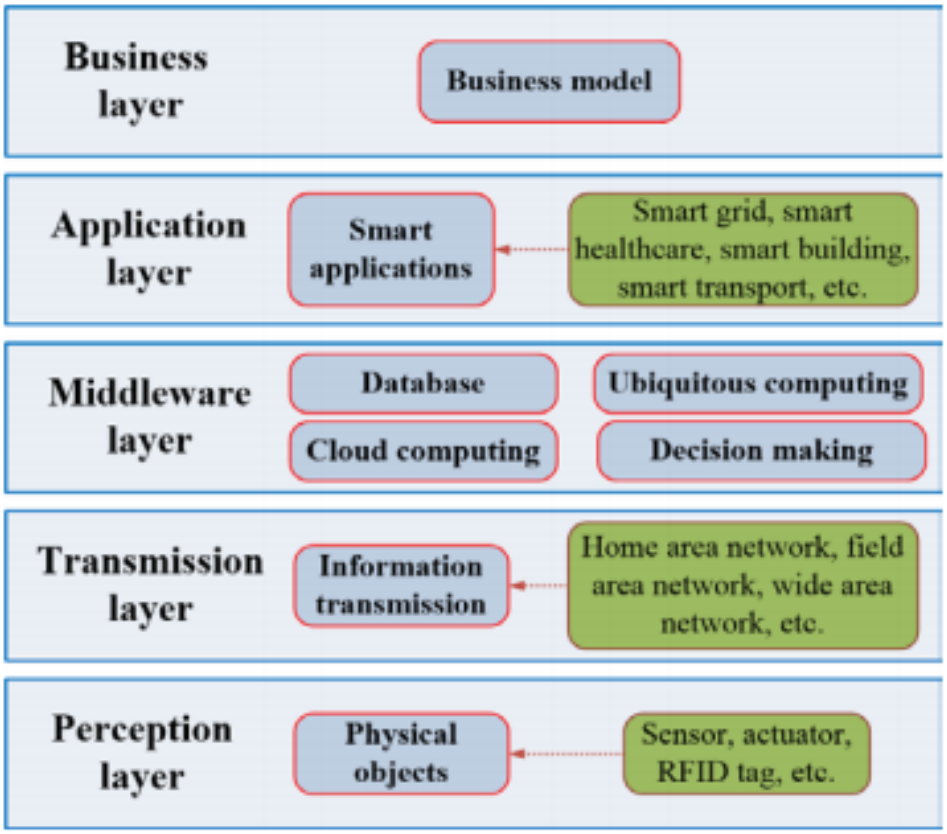


Figure 1.5 Architecture à cinq couches [64]

Le tableau suivant (Tableau 1.1) montre les différents protocoles IoT selon l'architecture à cinq couches :

Couches	**Protocoles**
Couche métier (buisness layer)	Gère l'ensemble du système IoT
Couche Application (Application layer)	• CoAP • MQTT • XMPP • AMQP • WEBSOCKET • DDS
La couche de traitement (processing layer**/** middleware layer)	La couche middleware (traitement) est une couche logicielle interposée entre la couche de transmission (transport) et couche d'application, qui peut associer directement les services aux demandeurs correspondants et a un lien vers la base de données. **La base de données**, l'informatique omniprésente, le cloud computing et la prise de décision peuvent avoir lieu dans cette couche [65].
Couche Transport / Couche réseau (transport layer/network layer)	• UDP : protocole de couche de transport simple pour les applications réseau client / serveur basé sur le protocole Internet (IP).UDP est souvent utilisé dans des applications spécialement conçues pour des performances en temps réel. • TCP : Protocole utilisé pour la majorité des connexions Internet. • DTLS:Le protocole DTLS assure la confidentialité des communications pour les protocoles de datagramme. Le protocole permet aux applications client / serveur de communiquer d'une manière conçue pour empêcher l'écoute clandestine. Le protocole DTLS est basé sur le protocole de transport Protocole Layer Security (TLS) et offre des garanties de sécurité équivalentes.
	• RPL : RPL (routage IPv6 pour les réseaux à faible puissance / perte) • 6LOWPAN : 6LoWPAN est un acronyme d'IPv6 sur les réseaux personnels sans fil à faible puissance. Il s'agit d'une couche d'adaptation pour IPv6 sur les liaisons IEEE802.15.4. Ce protocole fonctionne uniquement dans la gamme de fréquences 2,4 GHz avec un taux de transfert de 250 kbps.
Couche Perception	• IEEE 802.15.4 : une norme qui spécifie la couche physique et le contrôle d'accès au support pour les réseaux personnels sans fil à bas débit (LR-WPAN). Il est géré par le groupe de travail IEEE 802.15.

Tableau 1.1 Protocoles de chaque couche [48]

I.2.7 Caractéristiques d'une application de IOT

Les caractéristiques d'une application IoT sont à l'origine du succès de l'IoT. Chaque caractéristique englobe un ensemble de fonctionnalités qui font de l'IoT un succès. Parmi les caractéristique de l'IoT il y'a :

- **La connectivité:** la connectivité est la caractéristique principale de l'IoT car elle permet au système d'envoyer des données et de rester connecté à d'autres appareils. Il fournit l'accessibilité au réseau du système et fonctionne en collaboration.[36]
- **L'efficacité énergétique, la qualité et la fiabilité:** les appareils de l'IoT peuvent fonctionner dans des conditions météorologiques extrêmes, des environnements impitoyables et des endroits difficiles d'accès, par exemple des appareils fonctionnant au cœur des mines. Étant donné que les appareils peuvent fonctionner dans de tels environnements, il est important qu'ils soient fabriqués avec la plus haute qualité, soient fiables et écoénergétiques.[33]
- **L'intelligence :** Les systèmes IoT sont très appréciés sur le marché en raison de leur intelligence. Une combinaison d'algorithmes et d'ordinateur permet au système d'informer le changement d'environnement et de prendre les mesures appropriées. Par exemple, les systèmes sont suffisamment intelligents pour détecter une pointe soudaine de température et déclencher une alarme d'incendie. [36]
- **La sécurité:** la sécurité est la caractéristique la plus importante de tout système. Si le système n'est pas sécurisé contre les cybers attaques et les interventions illégales, personne ne l'utilisera. Les systèmes IoT traitent des données personnelles, c'est pourquoi il est obligatoire que toutes les mesures de sécurité soient prises en compte dans ce système. Tous les systèmes IoT sont suffisamment sécurisés pour traiter les données personnelles. [36]
- **Les capteurs:** les capteurs sont un aspect important des appareils et des systèmes au sein de l'Internet des objets. Les capteurs surveillent, suivent et mesurent l'activité et les interactions d'un appareil, puis relaient ces informations via le Cloud. Certains exemples de tels capteurs incluent ceux qui surveillent la santé et la forme physique d'une personne ou des capteurs qui peuvent détecter si une porte a été ouverte dans votre maison ou même ceux qui surveillent les statistiques d'utilisation. [33]
- **Rentabilité:** les objets ou dispositifs qui nécessitent des capteurs pour relayer des informations ont besoin d'être diffusés en grande quantité pour être efficaces, par conséquent ils devront être rentables. Par exemple, s'il s'agit d'un capteur attaché à des produits alimentaires pour surveiller les dates de péremption, il devra être implémenté dans chaque produit. [33]

I.3 Conclusion

Dans ce chapitre, nous avons présenté l'Iot et on a montré que cette technologie a contribué de manière efficace au développement de notre quotidien et notre vie en générale, car elle est utilisée dans plusieurs et différents domaines dont on a cité quelques-uns.

On a aussi présenté le fonctionnement de l'IoT ou on a montré ses différents composants ainsi que son architecture et les différents protocoles de l'IoT et on a terminé ce chapitre par présenter les différentes caractéristiques de l'IoT.

Le chapitre suivant sera consacré à la communication dans l'IoT, on citera et définira quelque protocoles de communication, ou on se concentrera sur les protocoles de la couche application.

Chapitre 2 : La communication dans l'IoT

II. Chapitre 2 : La communication dans l'IoT

II.1 Introduction

La communication dans l'IoT ce fait à l'aide de différents protocoles et le type de protocole IoT à utiliser dépend de la couche et de l'architecture système sur laquelle les données doivent circuler.

Après avoir introduit le concept d'Internet of Things et son fonctionnement dans le premier chapitre. Ce chapitre sera consacré à la communication dans l'IoT, ou on présentera les différents protocoles de la couche application

II.2 La Couche application

Les protocoles de la couche application sont utilisés pour échanger des données entre les programmes s'exécutant sur les hôtes source et de destination. Il existe des protocoles de la couche application qui permettent la communication dans l'IoT (XMPP, DDS, MQTT, CoAP, WebSocket et AMQP)

II.2.1 Les protocoles de la couche application

Un protocole applicatif est un ensemble de règles définissant le mode de communication entre deux applications informatiques. Ils se basent sur les protocoles de transport (TCP/UDP) pour établir dans un premier temps des routes et échanger les données selon l'ensemble des règles du protocole applicatif. [10] Les protocoles applicatifs les plus utilisés sont :

A. XMPP (Protocole de messagerie et de présence extensible)

XMPP est un ensemble de protocoles standards ouverts de l'Internet Engineering Task Force (IETF) pour la messagerie instantanée, et plus généralement une architecture décentralisée d'échange de données. XMPP est également un système de collaboration en quasi-temps-réel et d'échange multimédia par son extension Jingle, dont la voix sur réseau IP (téléphonie sur Internet), la visioconférence et l'échange de fichiers sont des exemples d'applications.

XMPP est constitué d'un protocole TCP/IP basé sur une architecture client-serveur permettant les échanges décentralisés de messages instantanés ou non, entre clients, au format Extensible Markup Language (XML). [31]

- **fonctionnement :**

Le protocole XMPP prend en charge les requêtes / réponses et publier / abonner des modèles; demande / réponse qui permet les communications bidirectionnelles et le modèle de publication/abonnement qui permet une communication multidirectionnelle (push et extraire les données). Une évolutivité élevée dans XMPP est fournie par architecture décentralisée. Il existe de nombreuses extensions pour Protocole XMPP, cela lui permet de travailler sur l'environnement sans infrastructure. [41]

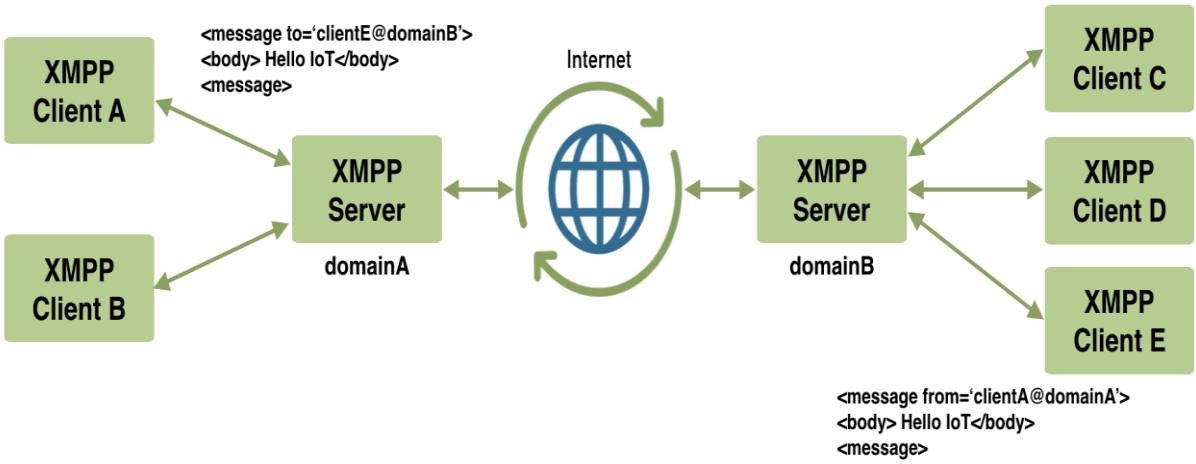


Figure 2.1 Fonctionnement du protocole XMPP [10]

B. DDS (Data Distribution Service)

Protocole de communication pair à pair polyvalent qui fait tout, de l'exécution de petits appareils à la connexion de réseaux hauts performances. DDS rationalise le déploiement, renforce la fiabilité et réduit la complexité. [12]

- **fonctionnement :**

DDS transmet des données entre l'éditeur et l'abonné comme sujet appelé. Les données sur le sujet sont générées par DataReader dans l'abonné et DataWritter dans les éditeurs. [43]

Figure 2.2 Modèle de communication DDS [44]

C. MQTT (Message Queuing Telemetry Transport):

MQTT est un Protocole de messagerie conçu pour une communication machine à machine légère, et principalement utilisé pour les connexions à faible bande passante vers des emplacements distants. MQTT utilise un modèle éditeur-abonné et est idéal pour les petits appareils qui nécessitent une utilisation efficace de la bande passante et de la batterie. [12]

- **fonctionnement :**

Un client, appelé publisher, établi dans un premier temps une connexion de type 'publish' avec le serveur MQTT, appelé broker. Puis, le publisher transmet les messages au broker sur un canal spécifique, appelé topic [10].

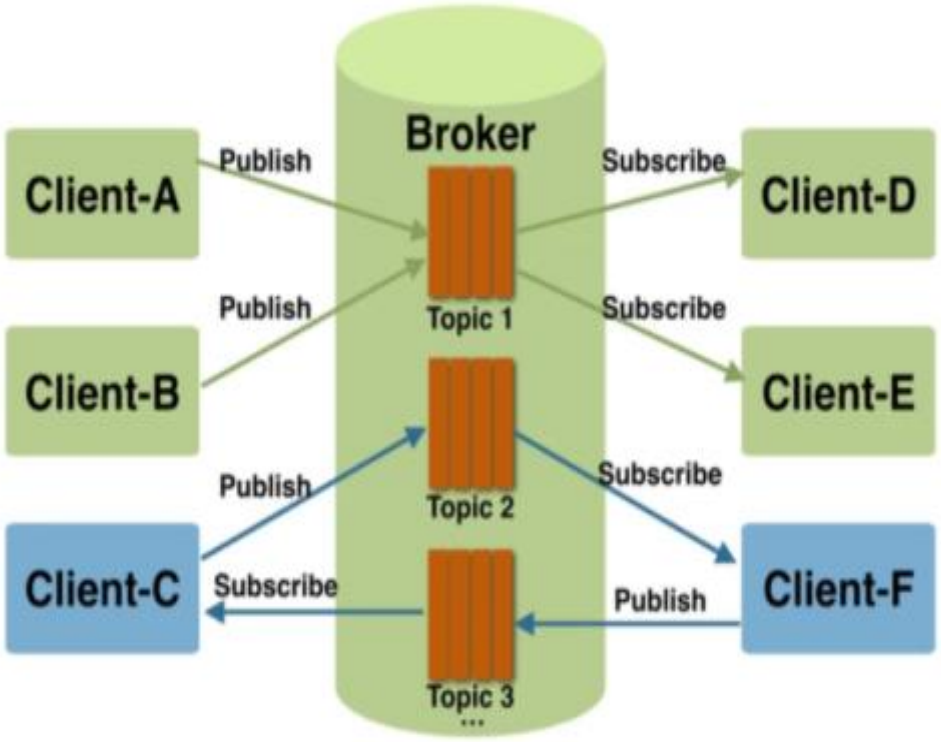


Figure 2.3 Fonctionnement du protocole MQTT [10]

D. CoAP (Constrained Application Protocol):

Protocole optimisé pour les bandes passantes et réseaux contraints, et conçu pour les appareils dont la capacité de connexion est limitée dans le cadre d'une communication machine à machine. CoAP est également un protocole de transfert de documents qui s'exécute sur le protocole UDP (User Datagram Protocol).[12]

- **fonctionnement :**

Pour transmettre une donnée, un client envoie à un serveur une requête CoAP, dans laquelle se trouve : le type du message, l'identifiant du message (mid) et une action (GET, POST, PUT ou DELETE) après le serveur va retourner une réponse [10].

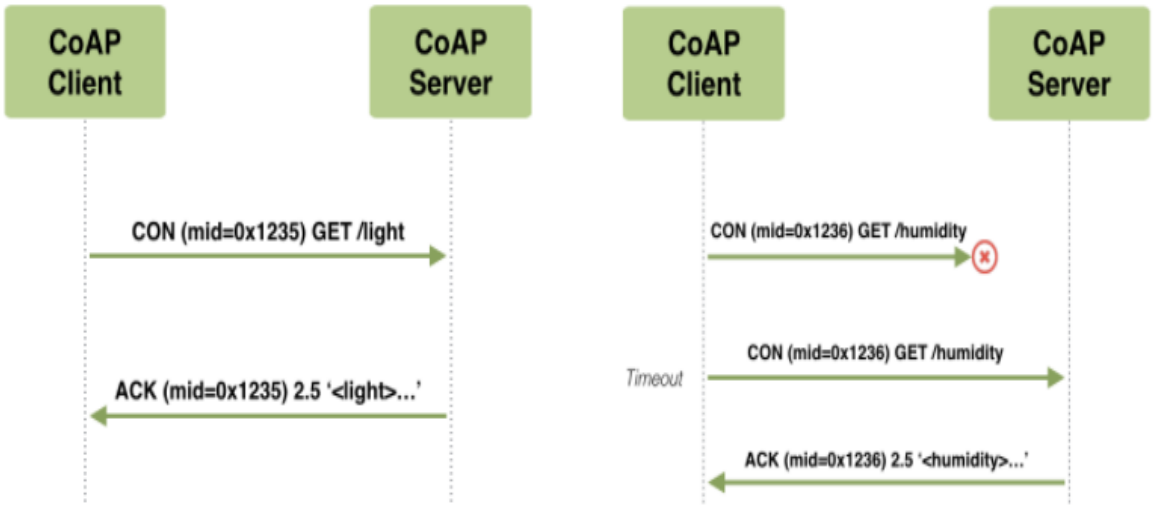


Figure 2.4 Fonctionnement du protocole CoAP [10]

E. WebSocket :

Le protocole Web-Socket offre deux méthodes pour communication entre les clients et un serveur distant. WebSocket fournit une sécurité similaire au modèle de sécurité utilisé Protocole HTTPS. Pour parcourir la couche d'application utilisée et Web-socket fonctionne sur le protocole de couche de transport TCP, besoin d'interagir et de communiquer avec l'hôte ceux qui se connectent avec télécommande. Web-Socket est un protocole Web qui fonctionne sur le canal TCP unique et fournit un duplex intégral les communications. Web-socket démarre la session sans modèles de publication / abonnement et demande / réponse comme les précédents protocoles [49].

- **Fonctionnement [10]:**

Le protocole Websocket permet l'établissement d'un canal de communication full-duplex en une seule connexion TCP entre un client et un serveur. La figure ci-dessous (figure 2.5) indique les trois principales phases de la vie du canal :

La phase de connexion appelé « Handshake » initié par le client
La phase d'échange bidirectionnel de messages
La phase de clôture du canal initié par l'une des deux parties

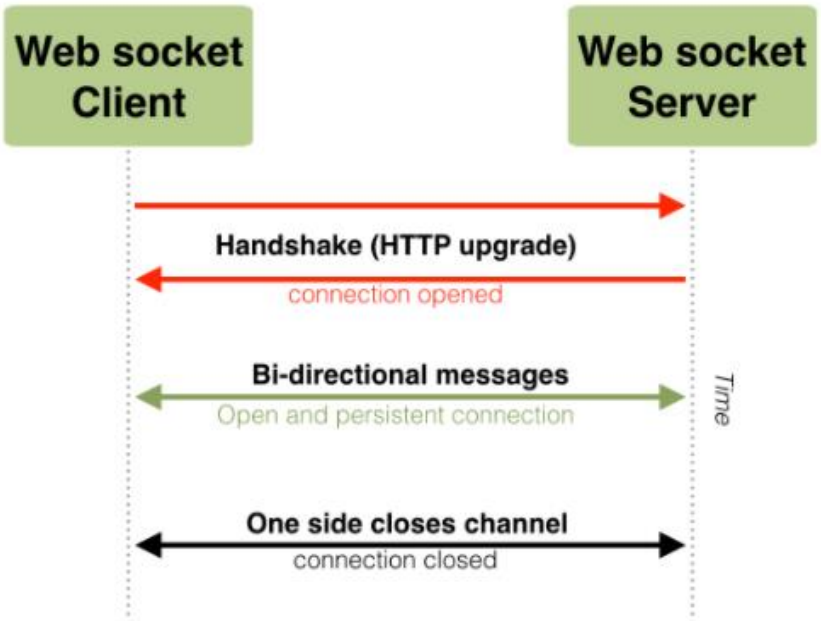


Figure 2.5 Fonctionnement de WebSocket[10]

F. AMQP (Advanced Message Queuing Protocol) :

AMQP est un protocole qui fait face à l'industrie financière. La sécurité est gérer avec l'utilisation des protocoles TLS/SSL. C'est écrasé TCP. AMQT suit la communication de publication / abonnement protocole de messagerie.[49]

- **Fonctionnement :**

Le fonctionnement du protocole AMQP est basé sur le même principe que celui de MQTT, toutefois la notion de *publisher*/*subsciber* est remplacée par celle de *producer*/*consumer*. En outre, grâce à un mécanisme interne noté « exchange », AMQP permet de router un message d'un producer vers plusieurs topics. Les critères de routage peuvent se faire de plusieurs façons ; inspection du contenu, de l'en-tête, clés de routage, etc. Ainsi, un même message peut être consommé par différents consumers via plusieurs topics. [10]

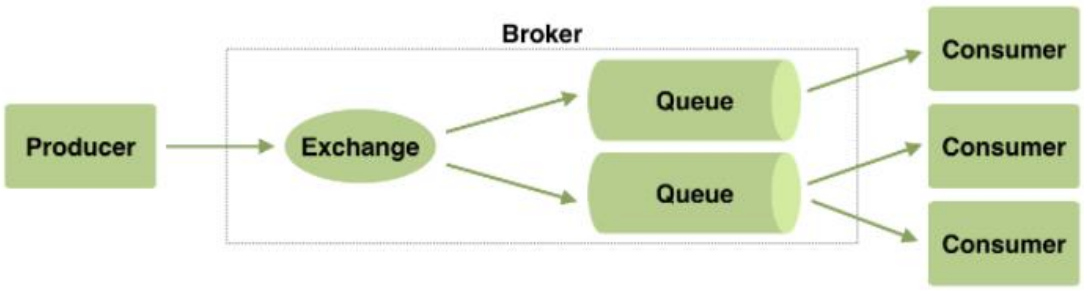


Figure 2.6 Fonctionnement du protocole AMQP [10]

II.3 Comparaison des protocoles applicatif

Dans cette partie nous allons faire une comparaison entre les protocoles applicatifs cités précédemment selon les différents critères suivant [32] :

- **Le standard du protocole**.
- **Le modèle de communication :** Deux types de communications :

  1. Communication publication / abonnement : ou les utilisateurs peuvent s'abonner pour un contenu précis.

  2. Communication requête / réponse : Dans cette communication un utilisateur peut acquérir des données avec des messages de requêtes personnalisés.
- **Le protocole de transport :** Protocoles de la couche transport :

  1. TCP : ou une connexion est nécessaire

  2. UDP : La connexion n'est pas nécessaire
- **La sécurité :** Soit :

  1. SSL(Secure Sockets Layer) /TLS (Transport Layer Security) : Transport Layer Security (TLS) et son prédécesseur Secure Sockets Layer (SSL), sont des protocoles de sécurisation des échanges par réseau informatique, notamment par Internet.[39]

  2. DTLS (Datagram Transport Layer Security) : Datagram Transport Layer Security (DTLS) est un protocole conçu pour protéger les données privées en prévenant la falsification, les écoutes, et la contrefaçon dans les communications. Il est basé sur le Transport Layer Security (TLS), qui est un protocole qui sécurise les réseaux de communications informatiques.[40]
- **Les principaux framework**
- **Le type de protocole :** il existe trois types de protocole : protocole de messagerie, de transfert web et protocole réseau

Le tableau suivant montre les différences qui peuvent avoir entre les protocoles IoT selon les caractéristiques que nous avons présentées [32],[10]:

	XMPP	DDS	MQTT	COAP	AMQP	WebSocket
Standard	L'Internet Engineering Task Force (IETF)	Object Management Group (OMG)	Organization for the Advancement of Structured Information Standards (OASIS)	L'Internet Engineering Task Force (IETF)	Organization for the Advancement of Structured Information Standards (OASIS)[51]	L'Internet Engineering Task Force (IETF)
Modèle de communication	Requête/ Réponse Publication/ Abonnement	Publication/ Abonnement	Publication/ Abonnement	Requête/ Réponse	Publication/ Abonnement	bidirectionnel
Protocole de transport	TCP	TCP/UDP	TCP	UDP	TCP	TCP
Sécurité	TLS/SSL	DTLS/SSL	TLS/SSL	DTLS	TLS/SSL	TLS/SSL
Framwork	Jabber, XMPPFramework		Emqtt, HiveMQ, Mosquitto, Eclipse Paho	Eclipse Californium, nCoAP	RabbitMQ, StormMQ	jetty websocket, Apache Tomcat
Type de protocole	Messagerie	Messagerie	Messagerie	Transfert web	Messagerie	Réseau

Tableau 2.1 Tableau comparatif de différents protocoles de la couche application

Apres avoir présenté les différents protocoles IoT de la couche application et donner quelques différences nous avons vu que chaque protocole a ces propre caractéristiques qui peuvent répondre à un besoin dans l'IoT et les deux protocoles qui nous ont le plus intéressé sont les protocoles CoAP et MQTT du à leur différentes performance par exemple le protocole CoAP Rapide et efficace grâce à la dépendance à l'UDP à faible coût, Et pour MQTT l'architecture du Broker peut simplifier la gestion; TCP et les options de qualité de service permettent une distribution robuste des messages.

II.4 Conclusion

Dans ce chapitre nous avons cité les différents protocoles de la couche application et nous avons donné leur caractéristique ainsi qu'une comparaison entre les différents protocoles applicatif les plus connu, nous avons aussi éclairé quelque points essentiels des principes de fonctionnement de chaque protocole.

Le prochain chapitre, sera consacré à une étude par simulation entre les deux protocoles CoAP et MQTT, afin de comparer leurs efficacités et leurs performances.

Chapitre 3 : protocoles MQTT et CoAP

III. Chapitre 3 : protocoles MQTT et CoAP

III.1 Introduction

COAP et MQTT sont les deux protocoles de communication les plus utilisés dans l'Internet des objets, ils sont souvent utilisé par les périphériques à ressources limitées.

On a déjà donné une brève définition des deux protocoles dans le chapitre précédent. Dans ce chapitre on va expliquer en détailler la structure et le fonctionnement de ces deux protocoles.

III.2 Le protocole MQTT

III.2.1 Historique

MQTT a été créé en 1999 par deux ingénieurs - Andy Stanford-Clark (IBM) et Arlen Nipper (Eurotech). La conception de MQTT a été motivée par la création d'un protocole léger et économe en bande passante, indépendant des données et prenant en charge plusieurs niveaux de qualité de service (QoS). Fait intéressant, même aujourd'hui, ce sont les mêmes raisons pour lesquelles MQTT est choisi pour mettre en œuvre des solutions IoT. En 2011, IBM et Eurotech ont fait don de MQTT au projet Eclipse proposé appelé Paho. En 2013, il a été soumis à l'OASIS pour normalisation. La dernière version de la spécification de protocole, 3.1.1 est devenue une norme OASIS. [15] Il devient un standard ISO (ISO/IEC 20922) en 2016. [16]

III.2.2 Mode de fonctionnement

Contrairement au principe du client/serveur utilisé sur le Web (figure 3.1), MQTT utilise celui de la publication/souscription ou plusieurs clients se connectent à un seul serveur (le broker) et ne communiquent pas entre eux ils vont soit publié des informations, ou souscrire à leur réception (figure 3.2). Les messages sont envoyé par des publieurs (publisher) sur un canal appelé topic ces messages peuvent être lu par les abonnés ou les souscripteurs (subscriber) les topics aux canaux d'information peuvent avoir une hiérarchie qui permet de sélectionner les informations que l'on désire.

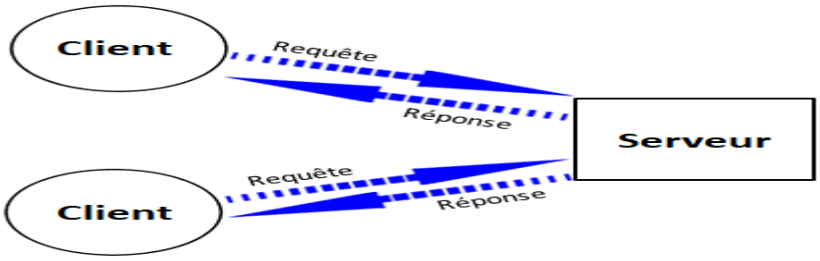


Figure 3.1 Principe Client/Serveur

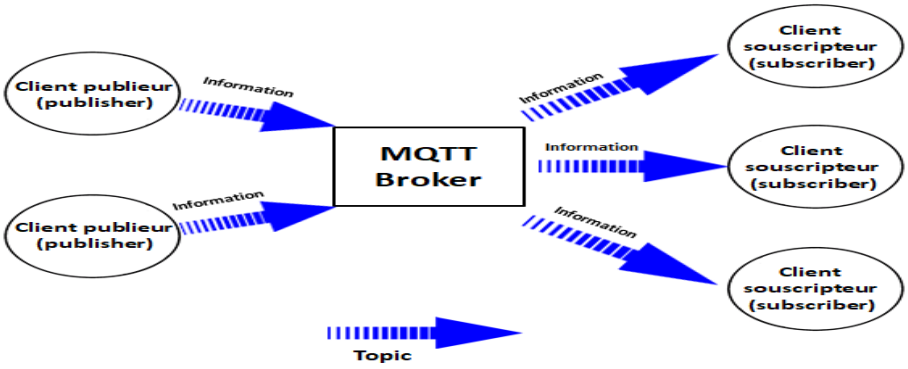


Figure 3.2 Principe du fonctionnement du protocole MQTT

III.2.2.1 Composants MQTT

MQTT se compose de :

a. le client : un client dans MQTT peut être soit subscriber (souscripteur) ou publisher (publieur)

b. Broker : le serveur qui gère la transmission des données entre les clients.

c. Topic : Un topic dans MQTT ou ce qu'on appelle un sujet est un point de terminaison ou les clients se connectent. c'est un centre de distribution central pour la publication et l'abonnement des messages.

d. le message ou l'information : C'est l'information qu'on souhaite échanger entre les appareils. elle peut être soit une commande ou des données.

III.2.2.2 Fonctionnement

A. Connexion et Déconnexion du client

- **la connexion :** Tout d'abord le client doit s'enregistrer auprès du broker et cela avec la commande CONNECT pour permettre l'échange des paramètres de connexion telle que les identifiants du client, le broker va soit confirmer au client qu'il est bien inscrit ou il va indiquer qu'une erreur a été détecté en renvoyant un code d'erreur de retour et un CONNACK. Pour que le broker soit au courant que le client est toujours actif il faut utiliser la commande PINGREQ et le broker va lui répondre avec un PINGRESP pour lui indiquer que la connexion est toujours active.
- **la déconnexion :** Si le client souhaite se déconnecter il envoie une commande DISCONNECT au broker.

B. L'abonnement et le désabonnement

- **l'abonnement :** Les clients s'enregistrent avec la commande SUBSCRIBE auprès du broker sur des topics qui seront des chemins d'accès aux ressources. Les clients recevront une notification lorsque quelqu'un publie sur ces topics.
- **Désabonnement :** Si un client souhaite annuler un abonnement d'un ou plusieurs topics il utilise la commande UNSUBSCRIBE et ainsi il ne recevra plus les publications qui concernent ces topics. La bonne réception de cette commande est confirmée par le broker par un UNSUBACK portant le même identifiant de paquet.
- **les topics (sujets) [17] :** Dans MQTT, le mot rubrique fait référence à une chaîne UTF-8 que le courtier utilise pour filtrer les messages pour chaque client connecté. Le sujet comprend un ou plusieurs niveaux de sujet. Chaque niveau de sujet est séparé par une barre oblique (figure 3.3).

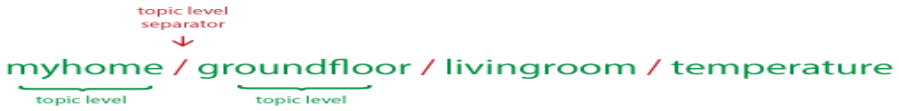


Figure 3.3séparateur de niveau de sujet

Par rapport à une file d'attente de messages, les rubriques MQTT sont très légères. Le client n'a pas besoin de créer le sujet souhaité avant de le publier ou de s'y abonner. Le broker accepte chaque sujet valide sans aucune initialisation préalable. Voici quelques exemples de sujets:

Myhome/ez-de-chaussée/salon/température
Allemagne / Bavière / voiture / 2382340923453 / latitude

Notez que chaque rubrique doit contenir au moins 1 caractère et que la chaîne de rubrique autorise les espaces vides. Les sujets sont sensibles à la casse.

Par exemple :

_myhome / temperature et _MyHome / Temperature sont deux sujets différents.

De plus, la barre oblique seule est un sujet valide Il est possible de définir une arborescence à l'aide du séparateur /.

Lorsqu'un client s'abonne à un sujet, il peut s'abonner au sujet exact d'un message publié ou il peut utiliser des caractères génériques pour s'abonner à plusieurs sujets simultanément. Un caractère générique ne peut être utilisé que pour s'abonner à des sujets, pas pour publier un message. Il existe deux types différents de caractères génériques: à niveau-unique et à multi-niveaux.

- **Niveau unique: +** Comme son nom l'indique, un caractère générique à un niveau remplace un niveau de sujet. Le symbole plus représente un caractère générique à un niveau dans une rubrique.
- **Multi Level: #** Le caractère générique à plusieurs niveaux couvre de nombreux niveaux de sujet. Le symbole de hachage représente le caractère générique à plusieurs niveaux dans le sujet. Pour que le courtier détermine les sujets qui correspondent, le caractère générique à plusieurs niveaux doit être placé en tant que dernier caractère du sujet et précédé d'une barre oblique.

C. La publication:

La commande PUBLISH permet aux abonnés de publier un message qui sera transmis par le broker aux abonnés éventuels. La même commande sera envoyée par le broker aux abonnés pour délivrer le message.

III.2.3 La QoS dans le MQTT

Parmi les caractéristiques du protocole MQTT il ya La QoS et c'est une caractéristique clé car La QoS offre au client la possibilité de choisir un niveau de service qui correspond à la fiabilité de son réseau et à sa logique d'application. Parce que MQTT gère la retransmission des messages et garantit la livraison.

Trois niveaux de qualité de service (QoS) sont définis pour la publication des messages [18] :

- **QoS 0:** Livraison une fois maximum.

Les messages sont envoyés en fonction des capacités maximum du réseau TCP/IP sous-jacent. Aucune réponse n'est attendue. Aucune sémantique de relance n'est définie dans le protocole. Par conséquent, le message ne parvient pas du tout ou une seule fois au broker de destination. Le niveau QoS 0 est également connu comme fire and forget.

- **QoS 1:** Livraison au moins une fois.

L'arrivée d'un message QoS 1 au broker est reconnue. En cas d'échec identifiable de la liaison ou de l'unité d'envoi, ou bien après une certaine période de temps sans réception du message d'accusé de réception, l'expéditeur envoie à nouveau une copie du message. Par conséquent, le message est sûr d'arriver, mais il peut le faire plusieurs fois.

- **QoS 2:** Livraison exactement une fois.

Pour le niveau QoS 2, d'autres flux de protocoles sont utilisés au-delà de QoS 1 pour que des messages ne soient pas envoyés en double à l'application de réception. Il s'agit du niveau de service le plus élevé qui sert lorsque des messages en double ne sont pas appropriés. Il existe

évidemment des conséquences en matière de trafic réseau, mais cet impact est souvent acceptable sachant l'importance du contenu du message.

La qualité de service peut être sélectionnée message par message, ce qui permet la publication des messages d'importance mineure avec le niveau QoS 0 et la distribution des messages plus importants avec QoS 2.
Ces niveaux sont mis en œuvre par des échanges supplémentaires entre l'expéditeur et le récepteur, et plus la qualité demandée est élevée, plus il faudra d'échanges pour valider une publication.

III.2.4 Format de paquet de contrôle MQTT(1)

Un paquet de contrôle MQTT se compose de trois parties au maximum, comme illustré dans la figure 3.4.

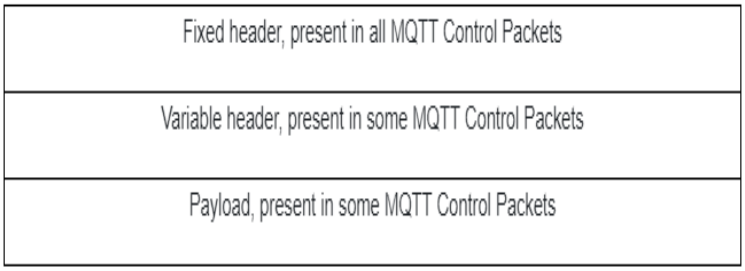


Figure 3.4 Structure d'un paquet de contrôle MQTT

III.2.4.1 Fixed header (En-tête fixe) :

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type				Flags specific to each MQTT Control Packet type			
byte 2...	Remaining Length							

Figure 3.5 Format d'en-tête fixe

(1) Tous les tableaux de cette section sont extraits du ref [20]

- **les composants de l'en-tête :**

A. MQTT Control Packet type (Type de paquet de contrôle MQTT):

Name	Value	Direction of flow	Description
Reserved	0	Forbidden	Reserved
CONNECT	1	Client to Server	Client request to connect to Server
CONNACK	2	Server to Client	Connect acknowledgment
PUBLISH	3	Client to Server/Server to Client	Publish message
PUBACK	4	Client to Server/Server to Client	Publish acknowledgment
PUBREC	5	Client to Server/Server to Client	Publish received (assured delivery part 1)
PUBREL	6	Client to Server/Server to Client	Publish release (assured delivery part 2)
PUBCOMP	7	Client to Server/Server to Client	Publish complete (assured delivery part 3)
SUBSCRIBE	8	Client to Server	Client subscribe request
SUBACK	9	Server to Client	Subscribe acknowledgment
UNSUBSCRIBE	10	Client to Server	Unsubscribe request
UNSUBACK	11	Server to Client	Unsubscribe acknowledgment
PINGREQ	12	Client to Server	PING request
PINGRESP	13	Server to Client	PING response
DISCONNECT	14	Client to Server	Client is disconnecting
Reserved	15	Forbidden	Reserved

Tableau 3.1 Type des paquets MQTT

B. Flags specific to each MQTT Control Packet type:

Les bits restants [3-0] de l'octet 1 dans l'en-tête fixe contiennent des indicateurs spécifiques à chaque type de paquet de contrôle MQTT, comme indiqué dans le Tableau 3.2 ci-dessous.

Control Packet	Fixed header flags	Bit 3	Bit 2	Bit 1	Bit 0
CONNECT	Reserved	0	0	0	0
CONNACK	Reserved	0	0	0	0
PUBLISH	Used in MQTT 3.1.1	DUP[1]	QoS[2]	QoS[2]	RETAIN[3]
PUBACK	Reserved	0	0	0	0
PUBREC	Reserved	0	0	0	0
PUBREL	Reserved	0	0	1	0
PUBCOMP	Reserved	0	0	0	0
SUBSCRIBE	Reserved	0	0	1	0
SUBACK	Reserved	0	0	0	0
UNSUBSCRIBE	Reserved	0	0	1	0
UNSUBACK	Reserved	0	0	0	0
PINGREQ	Reserved	0	0	0	0
PINGRESP	Reserved	0	0	0	0
DISCONNECT	Reserved	0	0	0	0

Tableau 3.2 Description du drapeau (flag) dans l'en-tête fixe du protocole MQTT

C. Remaining Length :

La longueur restante est le nombre d'octets restant dans le paquet actuel, y compris les données dans l'en-tête variable et la charge utile. La longueur restante n'inclut pas les octets utilisés pour coder la longueur restante.

La longueur restante est codée à l'aide d'un schéma de codage de longueur variable qui utilise un seul octet pour des valeurs allant jusqu'à 127. Les valeurs plus grandes sont traitées comme suit. Les sept bits les moins significatifs de chaque octet codant les données et le bit le plus significatif est utilisé pour indiquer qu'il y a des octets suivants dans la représentation. Ainsi, chaque octet code 128 valeurs et un "bit de continuation". Le nombre maximal d'octets dans le champ Longueur restante est de quatre.[20]

III.2.4.2 Variable header (en tête-variable) :

L'en-tête variable n'est pas présente dans tous les paquets MQTT. Certaines commandes ou messages MQTT utilisent ce champ pour fournir des informations ou des indicateurs supplémentaires et varient en fonction du type de paquet. Un identifiant de paquet est commun dans la plupart des types de paquets. 21]

III.2.4.3 Payload (charge utile) :

Au final, le paquet peut contenir une charge utile. Même la charge utile est facultative et varie en fonction du type de paquet. Ce champ est généralement contient les données envoyées. Par exemple pour le paquet CONNECT, la charge utile est l'ID client et «nom d'utilisateur et mot de passe» s'ils sont présents. Et pour PUBLISH packet, c'est le message à publier. [21]

III.2.4.4 CONNECT :

Le tableau 3.3 suivant est un exemple de paquet CONNECT [22]:

	Description
CONNECT packet fixed header	
byte 1	Control packet type
byte 2	Remaining length
CONNECT packet variable header	
Protocol name	
byte 1	Length MSB (0)
byte 2	Length LSB (4)
byte 3	(M)
byte 4	(Q)
byte 5	(T)
byte 6	(T)
Protocol level	
byte 7	Level (4)
Connect flags	
byte 8	CONNECT flags byte, see the table below for the bits.
Keep alive	
byte 9	Keep Alive MSB (0)
byte 10	Keep Alive LSB (60)
Client ID	
byte 11	Length MSB (0)
byte 12	Length LSB (4)

Tableau 3.3format d'un paquet CONNECT

Description des champs du paquet CONNECT [22]:

Champs	Description
Protocol Name	Le paquet de connexion commence par le nom du protocole, qui est MQTT. La longueur du nom du protocole (en octets) est immédiatement avant le nom lui-même.
Protocol Level	Fait référence à la version de MQTT utilisée, dans ce cas, une valeur de 4 indique la version 3.1.1 de MQTT.
Connect Flags	Indiquez certains aspects du paquet. Par souci de simplicité, cet exemple définit uniquement l'indicateur Clean Session, qui indique au client et au courtier de supprimer toute session précédente et d'en démarrer une nouvelle.
Keep Alive	La fréquence à laquelle le client envoie une requête ping au courtier pour maintenir la connexion active; dans cet exemple, il est défini sur 60 secondes.
Client ID	La longueur de l'ID (en octets) précède l'ID lui-même. Chaque client se connectant à un courtier doit avoir un ID client unique. Dans l'exemple, l'ID est DIGI. Lorsque vous utilisez les bibliothèques Paho MQTT Python, un ID alphanumérique aléatoire est généré si vous ne spécifiez pas d'ID.

Tableau 3.4 Champs du paquet CONNECT

III.2.4.5 CONNACK

Le paquet CONNACK est le paquet envoyé par le serveur en réponse à un paquet CONNECT reçu d'un client. Le premier paquet envoyé du serveur au client DOIT être un paquet CONNACK.

Si le client ne reçoit pas de paquet CONNACK du serveur dans un délai raisonnable, le client DEVRAIT fermer la connexion réseau. Un temps "raisonnable" dépend du type d'application et de l'infrastructure de communication.[20]

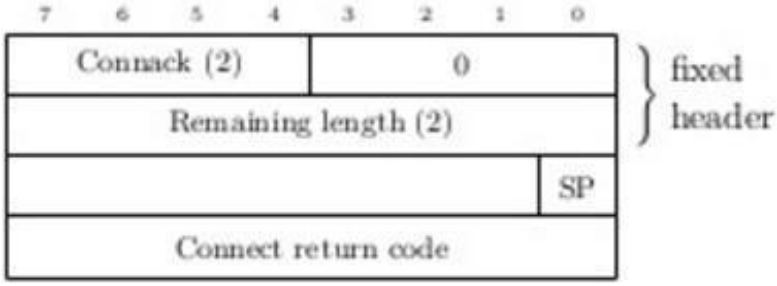


Figure 3.6 Format d'un paquet CONNACK [24]

III.2.4.6 PUBLISH

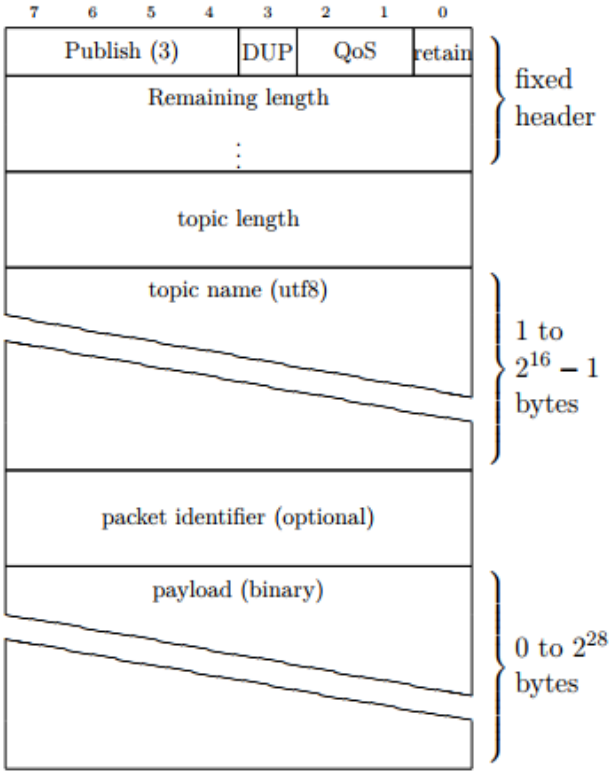


Figure 3.7 Format d'un paquet PUBLISH [24]

Le client expéditeur (publisher) a le choix d'envoyer des données binaires, des données texte ou même du XML etc. Un message PUBLISH dans MQTT à plusieurs attributs comme on peut voir dans l'exemple ci dessue :

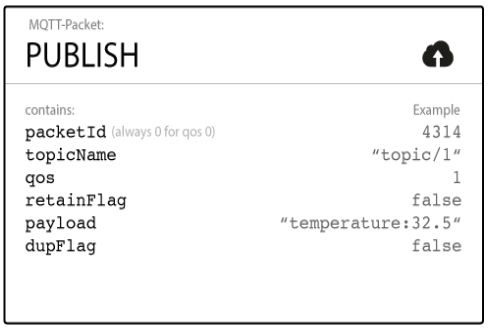


Figure 3.8 Attribut du paquet PUBLISH [24]

III.2.4.7 PUBACK, PUBREC, PUBREL, PUBCOMP :

Tous ces paquets ont la même structure, seules les valeurs du code de commande et du champ Reserved peuvent varier.

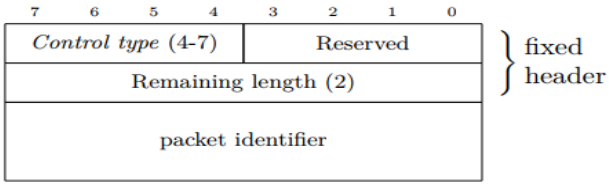


Figure 3.9 Format d'un paquet PUB[ACK/REC/REL/COMP]

III.2.4.8 SUBSCRIBE

Ce message d'abonnement est très simple, il contient un identifiant de paquet unique et une liste d'abonnements.

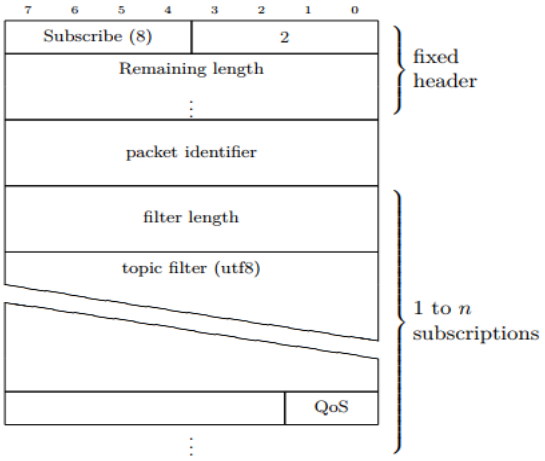


Figure 3.10 Format d'un paquet SUBSCRIBE [24]

III.2.4.9 SUBACK

Un paquet SUBACK contient une liste de codes de retour, qui spécifient le niveau de QoS maximal qui a été accordé dans chaque abonnement demandé par le SUBSCRIBE.[20]

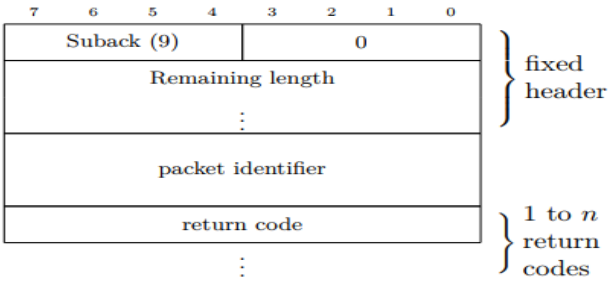


Figure 3.11 Format d'un paquet SUBACK [24]

III.2.4.10 UNSUBSCRIBE :

Le packet UNSUBCRIBE ressemble au paquet SUBSCRIBE :

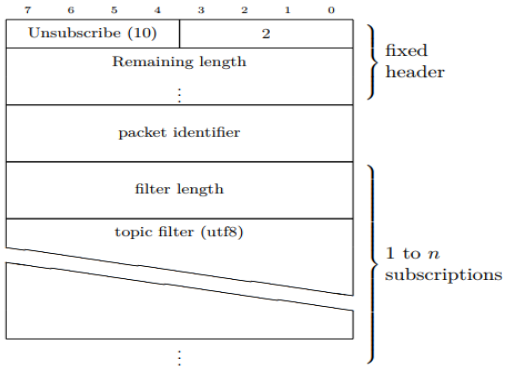


Figure 3.12 Format d'un paquet UNSUBSCRIBE [24]

III.2.4.11 UNSUBACK

UNSUBACK contient l'identifiant de paquet de la requête correspondante.

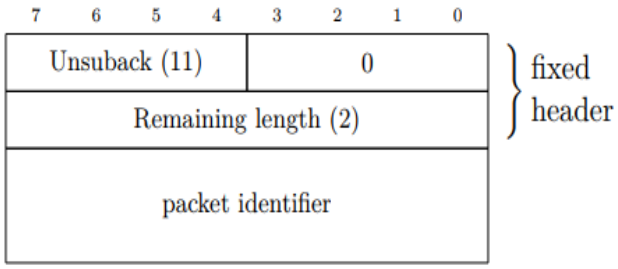


Figure 3.13 Format d'un paquet UNSUBACK [24]

III.2.4.12 PINGREQ, PINGRESP

Ces deux paquets sont constitués uniquement de l'entête fixe du protocole, tenant ainsi sur deux octets.

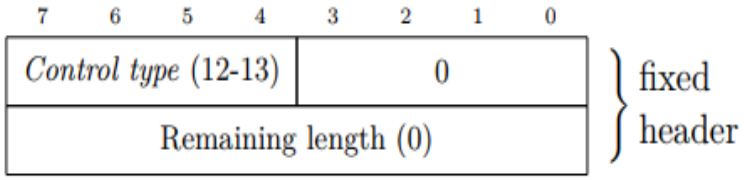


Figure 3.14 Format d'un paquet PING [REQ/RESP]

III.2.4.13 DISCONNECT

Comporte uniquement l'en tête fixe.

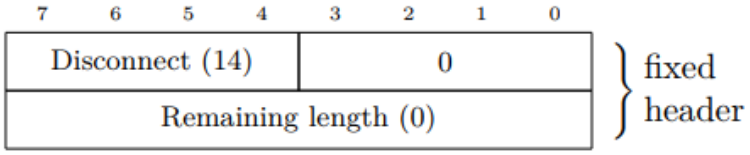


Figure 3.15 Format d'un paquet DISCONNECT

III.2.5 Exemple d'utilisation du protocole MQTT

A l'origine, le MQTT fut développé pour les liens de données à haute latence et faible bande passante utilisés dans l'industrie du gaz et du pétrole. Beaucoup d'autres applications fonctionnent avec ce système de transfert de données.
De nombreux objets connectés destinés aux particuliers, ainsi que des applications comme Facebook Messenger, reposent sur le MQTT. De même Amazon IoT se base sur ce protocole. En général, ce protocole est le plus adapté pour les systèmes de contrôle utilisés par les entreprises industrielles. Son taux d'adoption devrait continuer à augmenter dans les années à venir.[66]

Il existe plusieurs applications qui utilisent le protocole MQTT parmi eux :
Haptik[62]:

Haptik est une application de chat qui connecte les utilisateurs à leurs assistants personnels numériques en temps réel. Lorsque vous discutez avec votre assistant personnel, vous voulez le moins de retard possible afin de faire avancer les choses rapidement.

Haptik est passé de XMPP à MQTT, parmi les raisons de ce changement :

- MQTT Prend en charge QoS
- Son Mécanisme de Pub / Sub
- Faible consommation d'énergie

III.3 Le protocole CoAP

III.3.1 Historique

Le protocole CoAP modifie certaines Fonctionnalités HTTP pour répondre aux exigences de l'IOT telles que la faible Consommation d'énergie et le fonctionnement en présence de liens à perte et Bruyants. ce protocole a été conçu sur la base de REST qui représente un moyen plus simple d'échanger des données entre les clients et les serveurs via HTTP.[45]

III.3.2 Mode de fonctionnement

CoAP utilise un modèle client-serveur qui ressemble à celui de http figure (3.16), où les clients envoient des requêtes sur des ressources REST pour récupérer de l'information d'un capteur ou contrôler un périphérique et son environnement. Il faut noter que CoAP traite les échanges de manière asynchrone au travers de datagrammes UDP

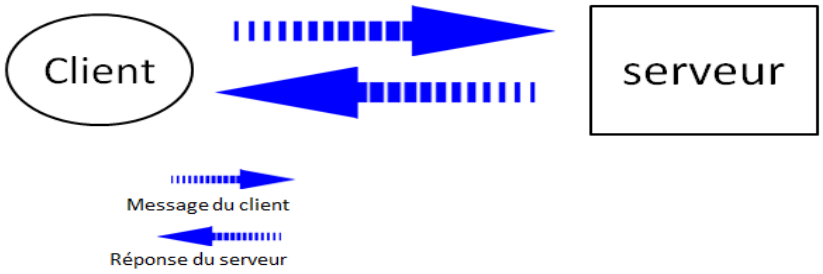


Figure 3.16 Principe du fonctionnement du protocole CoAP

III.3.2.1 Composants

a. **client :** le client envoi une requête qui est similaire à une requête http pour demander une action.
b. **serveur :** répond au client par un un code réponse.
c. **message :** les informations échangé entre le client et le serveur.

III.3.2.2 Fonctionnement

CoAP s'appuie sur une approche à deux couches, une couche de messagerie CoAP ou 4 message sont définis (tableau 3.5) et une couche d'interaction sous forme de requête/réponse héritée du protocole HTTP (figure 3.18)

Message	Description
CON	Confirmable: Le message requiert une réponse qui peut être véhiculée par un message d'acknowlegment ou envoyé de façon asynchrone si la demande ne peut être traitée immédiatement. Peut-être aussi bien une requête qu'une réponse qui doit être acquittée.
ACK	*Acknowledgment* : Le message confirme la réception d'un message *Confirmable*.
NON	*Non-confirmable* : Le message ne requiert aucune réponse ou acquittement.
RST	*Reset* : Dans le cas où le message n'a pu être traité.

Tableau 3.5 Messages de la couche messagerie CoAP

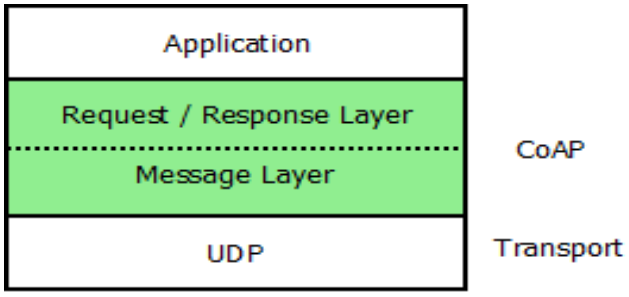


Figure 3.17 Architecture CoAP

La sémantique des requêtes et réponses COAP est transportée dans des messages COAP qui incluent soit un code de méthode (Tableau 3.5) ou un code de réponse (e.g., 404, 205, etc).

Un jeton (token) est utilisé pour associer une requête à une réponse indépendamment des messages qui les transportent. Le Token est indépendant de l'identificateur de message. [26]

Méthode	Description
GET	Cette méthode permet de récupérer des informations d'une ressource identifiée
POST	Permet de créer une nouvelle ressource dans l'URL demandé
PUT	Permet de mettre à jour la ressource dans l'URL demandé
DELETE	Cette méthode supprime la ressource dans l'URL demandé

Tableau 3.6 Les méthodes CoAP

Une requête est transportée dans un message CON (Confirmable) ou NON (Non Confirmable):

- Si la réponse est disponible immédiatement dans le cas d'un message CON, elle est transportée dans un message Acknowledgement (ACK).Une des réponses avec (2.05) indique un succès et l'échec avec (4.04) (figure 3.19)

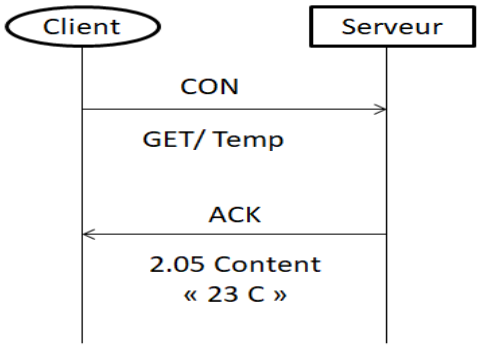


Figure 3.18réponse disponible dans le cas de message CON avec méthode GET

- Si le message ACK est perdu, l'émetteur du message CON le retransmettra.(figure 3.20)

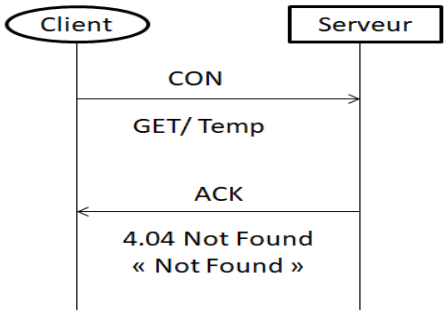


Figure 3.19 Dans le cas d'un ACK perdu

- Si le serveur n'est pas en mesure de répondre immédiatement à une requête transportée dans un message CON, il répond simplement via un message ACK sans contenu. Lorsque la réponse est prête, le serveur l'émet dans un nouveau message CON. (Figure 3.21)

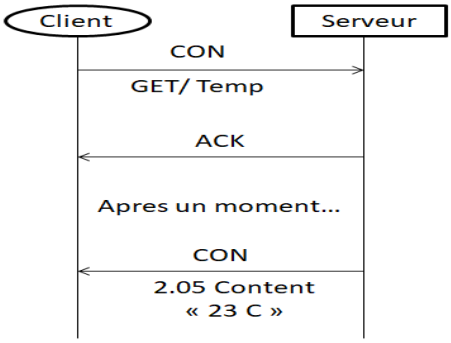


Figure 3.20 Serveur pas en mesure de répondre directement

- Si une requête est émise dans un message Non-confirmable (NON), alors la réponse est émise via un nouveau message NON. (Figure 3.22)

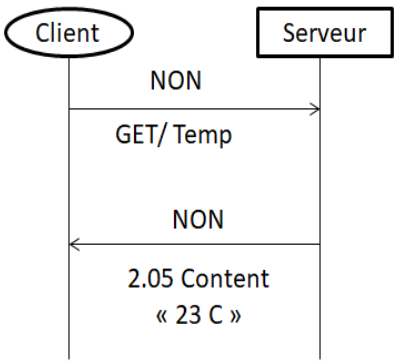


Figure 3.21 Message NON

Lorsqu'un récepteur n'est pas en mesure de traiter un message CON (même pas en mesure de fournir une réponse d'erreur), il répond avec un message Reset (RST) à la place du message Acknowledgement (ACK). (Figure 3.23)

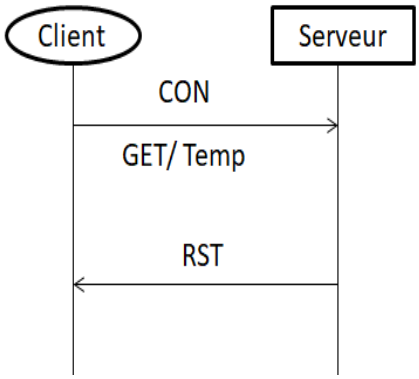


Figure 3.22 Message CON et RST

Lorsqu'un récepteur n'est pas en mesure de traiter un message NON, il peut répondre avec un message Reset (RST). (Figure 3.24)

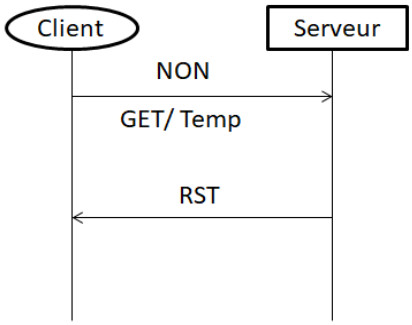


Figure 3.23 Message NON et RST

III.3.3 Message CoAP

L'en-tête CoAP a été conçu pour être facile à analyser par des programmes tournant sur de petits équipements comme les capteurs (3.25).

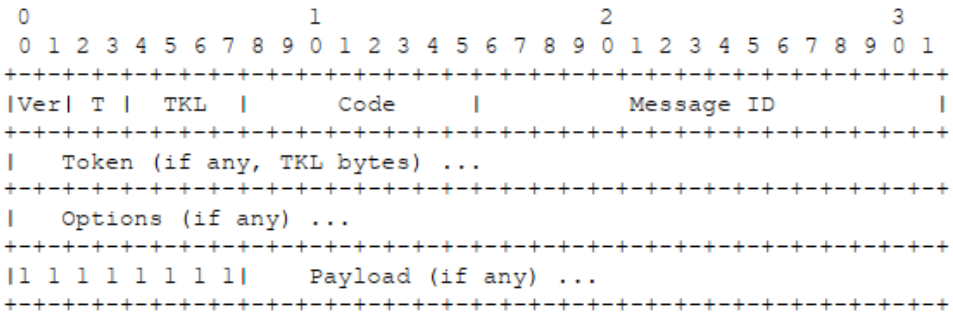


Figure 3.24 Format du message COAP [26]

III.3.3.1 L'En-tête

Au lieu du texte lisible de HTTP, l'en-tête de CoAP commence par une partie fixe de quatre octets, qui comprend [26] :

- **La version du protocole (Ver)** : il s'agit de la version du protocole CoAP
- **Le type de message (T):** Indique si le message est de type Confirmable (0), Non Confirmable (1), Acknowledgement (2), ou Reset (3).
- **La longueur du Token (TKL):** Indique la longueur (variable) du champ Token (0-8 octets) Les longueurs 9-15 sont réservées et ne doivent pas être utilisées. Si un récepteur reçoit une valeur comprise entre 9 et 15, il doit la traiter comme une erreur de format.
- **Code:** Entier sur 8 bits, séparé en 3 bits de class (bits les plus significatifs) et 5 bits de détail (bits les moins significatifs), documentés sous la forme c.dd ou « c » est un digit compris entre 0 et 7 (3 bits) et « dd » représente deux digits entre 00 et 31 (5 bits). La classe peut indiquer une requête (0), une réponse de succès (2), une réponse d'erreur client (4) ou une réponse d'erreur de serveur (5). Toutes les autres valeurs de classe sont réservées. Un cas particulier est celui d'un message vide dont le code est 0.00. Dans le cas d'une requête, le champ Code indique la méthode. (0.01 = GET; 0.02 = POST; 0.03 = PUT; 0.04 = DELETE).
- **Un Message ID sur deux octets** : Ce Message ID permet de détecter les duplicatas et d'associer un accusé de réception à un message précis. À noter qu'avec ces deux octets, on est limité à environ 250 messages par seconde (en raison du paramètre EXCHANGE_LIFETIME qui est à 247 secondes par défaut). Ce n'est pas une limite bien grave : les ressources des machines CoAP ne leur permettent pas d'être trop bavardes de toutes les façons.

III.3.3.1 Token

La longueur est indiquée par le champ TKL. La valeur de token est utilisée afin de corréler une requête et sa réponse. L 'en-tête et le Token sont suivis par zéro, une ou plusieurs options. [26]

III.3.3.2 Option

Une option peut être suivie par la fin du message, par une autre option ou un marqueur de payload (contenu) et le contenu lui-même.[26]

III.3.3.3 Payload

Le payload est optionnel. Si présent, il est préfixé par un marqueur de payload d'un octet (0xFF), qui indique la fin des options et le début du payload. Le payload commence après le marqueur et termine à la fin du datagramme UDP.

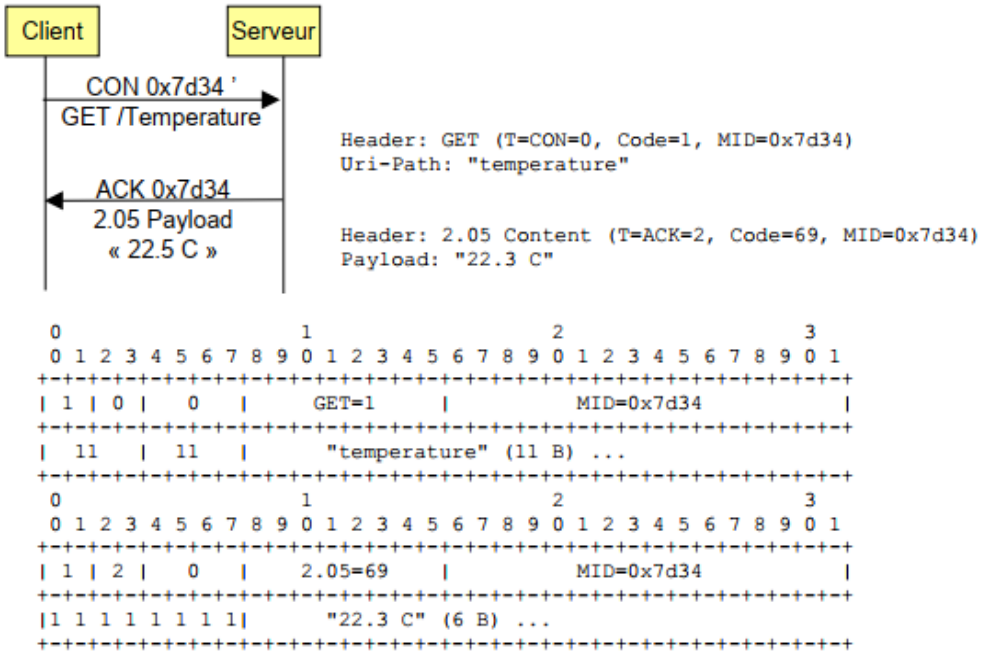


Figure 3.25 Exemple de requête et de réponse CoAP [26]

III.3.4 Exemple d'utilisation du protocole CoAP

Il existe des mises en œuvre de la domotique qui incluent plusieurs services comme la sécurité et le mesure de la température ambiante de la maison qui fonctionnent avec l'aide de CoAP

Au sein d'un réseau domestique, différents appareils IoT sont connectés et la communication est établie entre eux à l'aide du protocole CoAP. (Figure 3.26)

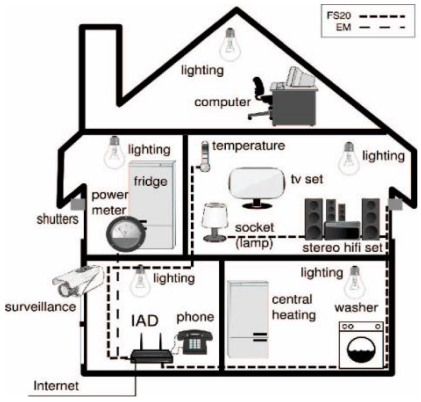


Figure 3.26 Exemple de configuration utilisant des protocoles propriétaires pour le contrôle de périphérique[63]

III.4 Différences entre MQTT et CoAP

	CoAP	**MQTT**
Protocole de couche de transport	Fonctionne sur UDP, TCP peut être utilisé	Fonctionne sur TCP, UDP peut être utilisé (MQTT-SN)
Mécanisme de fiabilité	CON, ACK, NON, RST	Les 3 niveaux de qualité de service
Modèle de communication	Demande/Réponse	Publication/Abonnement
En-tête	4 Octets	2 Octets
Mode de messagerie	Utilise à la fois le mode asynchrone et synchrone.	Utilise uniquement le mode asynchrone
Nombre de types de messages utilisés	4	16

Tableau 3.7différences majeures entre MQTT et COAP

III.4.1 Les points en commun entre les deux protocoles [67] :

- Ce sont des prometteurs pour les petits appareils
- Ils sont des standards ouverts
- Ils sont mieux adaptés aux environnements contraints que HTTP
- ils fournissent des mécanismes de communication asynchrone
- Ils sont exécuté sur IP
- Ils ont une gamme d'implémentations

III.5 Conclusion de comparaison

Apres avoir présenté le protocole CoAP et le protocole MQTT et expliqué le fonctionnement de chaque protocole on peut dire que les deux protocoles sont utile en tant que protocole de l'IoT mais avec des différences fondamentales.

Le protocole MQTT a une architecture qui se base sur la publication/abonnement et le Broker au milieu ce qui le rend idéal pour la communication sur un réseau étendu, ce protocole est aussi utile dans les cas où la bande passante est limite ce qui est pas mal.

Pour le protocole CoAP , ce qui le différencie de MQTT et le rend plus fort est ça compatibilité avec HTTP, il est construit sur le protocole UDP et ça c'est très utile pour l'utilisation de CoAP dans certain environnements a ressource limité, car il ne faut pas oublier que UDP permet la diffusion (Broadcast) et la multi diffusion (Multicast) donc la transmission peut se faire à plusieurs hôtes et ça en utilisant moins de bande passante ce qui rend ce protocole idéal pour les environnements de réseau local dans le quelle les périphériques doivent se parler rapidement.

III.6 Conclusion

Dans ce chapitre on a montré les différentes caractéristiques de MQTT et CoAP. Nous avons expliqué comment ces deux protocoles fonctionnent et on a montré les différences qui existent entre eux. Enfin nous avons conclu par un tableau de comparaison entre ces deux protocoles (MQTT et CoAP).

Dans le prochain chapitre nous allons opter a une étude comparative par simulation entre les deux protocole.

Chapitre 4 : Simulation

IV. Chapitre 4 : Simulation

IV.1 Introduction

Dans ce chapitre nous allons décrire le processus de la réalisation de notre simulation. Ceci en présentant l'environnement de développement, et une présentation de quelques interfaces de notre simulation, nous allons aussi proposer un scenario de simulation. Enfin nous allons évoluer les performances des protocoles MQTT et CoAP et les comparer, afin de conclure ou le mieux utiliser chacun deux.

IV.2 La Simulation

IV.2.1 Définition

La simulation informatique ou numérique désigne l'exécution d'un programme informatique sur un ordinateur ou réseau. Les simulations numériques scientifiques reposent sur la mise en œuvre de modèles théoriques. Elles sont donc une adaptation aux moyens numériques de la modélisation mathématique, et servent à étudier le fonctionnement et les propriétés d'un système modélisé ainsi qu'à en prédire son évolution. Les interfaces graphiques permettent la visualisation des résultats des calculs.[53]

IV.2.2 Les outils de simulation les plus connu dans le domaine de réseau :

Il existe plusieurs outils qui permettent de réaliser une simulation c'est ce qu'on appelle outils de simulation parmi eux :

- **NS2 (Network Simulator 2) :**NS2 est un simulateur à événements discrets qui permet d'exécuter tout type de scenarii sur des topologies définies par l'utilisateur. Pour visualiser les résultats, un outil graphique a été développé : NAM *(Network AniMator).* [27]
- **NS3 (Network Simulator 3) :**NS3 est un simulateur de réseau à événements discrets, principalement destiné à la recherche et à l'enseignement. ns-3 est un logiciel libre, sous licence GNU GPLv2, et est accessible au public pour la recherche, le développement et l'utilisation.[42]
- **NeSSi (Network Security Simulator) :** NeSSi est un outil de simulation de réseau qui intègre une variété de fonctionnalités pertinentes pour la sécurité du réseau.[28]
- **Omnet++ :** OMNeT ++ est une bibliothèque et un framework de simulation C ++ extensible, modulaire et basé sur des composants, principalement pour la construction de simulateurs de réseau. [52]
- **COOJA :** Cooja est un simulateur fourni avec Contiki. Il permet de simuler un réseau de capteurs. Cooja permet de simuler les connexions

réseaux et d'interagir avec les capteurs. Cet outil permet aux développeurs de tester les applications à moindre coût.

- **Le Boson NetSim Network Simulator :** est une application qui simule le matériel et les logiciels de mise en réseau de Cisco Systems et conçue pour aider l'utilisateur à apprendre la structure de commande de Cisco IOS. [68]

Afin d'étudier les protocoles de communication IoT, nous avons choisi pour notre simulation l'outil COOJA car il est open source, pratique et simple a utilisé et parmi les simulateurs les plus utiliser dans la simulation des protocoles des réseaux de capteurs sans fils (WSN)

IV.2.3 L'environnement d'évaluation :

Pour notre simulation on a choisi le simulateur COOJA car non seulement fournit des outils qui permettent de diffuser des données de chaque test très facilement et dans un format lisible et simple mais aussi il fournit une grande quantité de code d'exemple qui peut être utilisé.

IV.2.3.1 Contiki OS

Contiki est un système d'exploitation open source léger et flexible pour les nœuds de capteurs WSN créé par Adam Dunkels écrit en langage C. Contiki connecte de petits microcontrôleurs peu couteux et peu performant a internet. [54] [55]

IV.3 Scenario de simulation

IV.3.1 Pour le protocole CoAP

Le cas étudié concerne un système automatisé de contrôle de gaz dans une boulangerie qui produits environs 50.000 baguettes par jour , afin d'éviter les fuites de gaz qui peuvent provoquer des dégâts, la boulangerie est équiper d'un système de détection de gaz qui fonctionne avec le protocole CoAP pour cela dans la boulangerie il est installé des capteurs(des détecteurs) de gaz dans les coins susceptibles d'une éventuelles fuite (tuyau, four……) et lorsque une fuite est détecter une alarme se déclenche

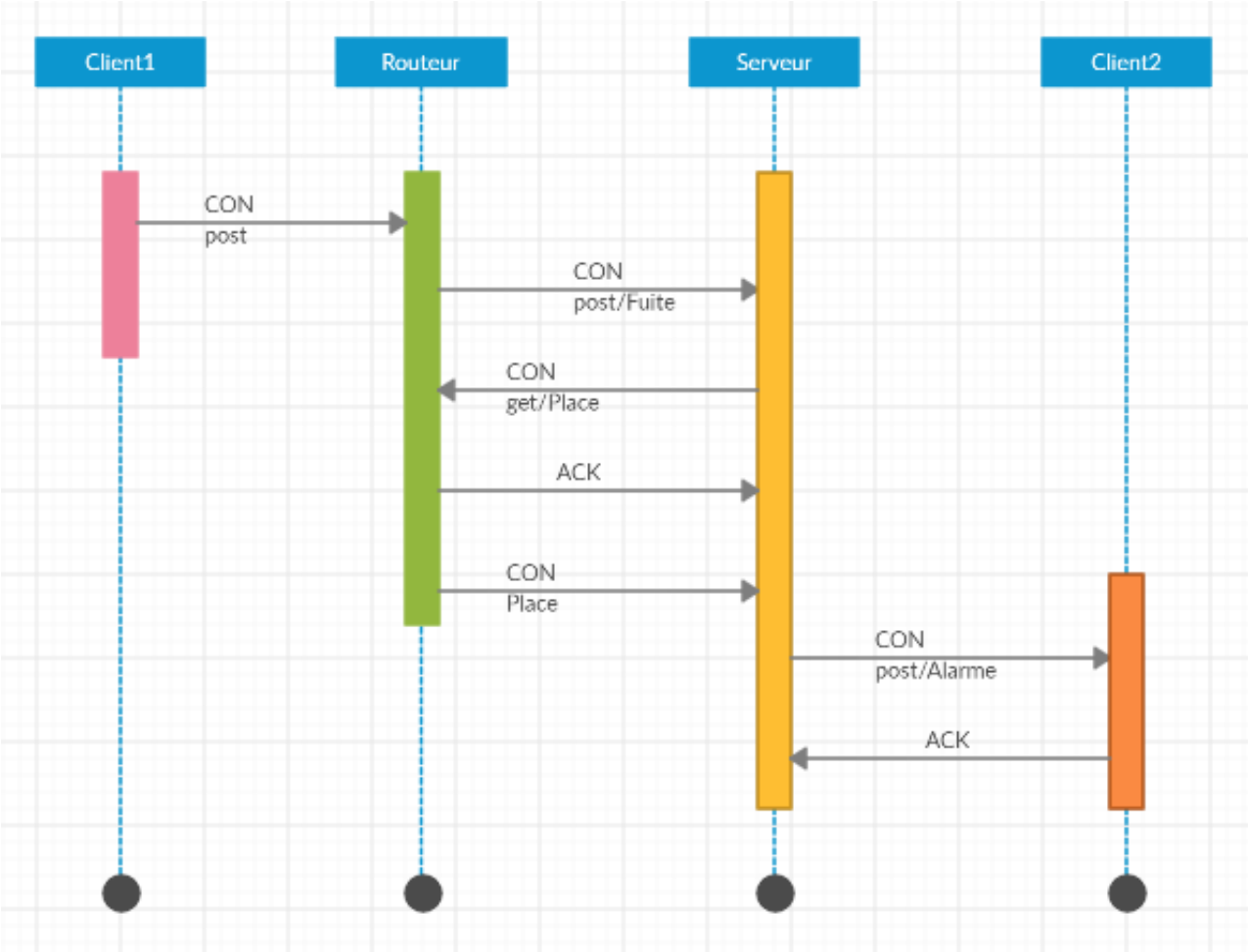
- **Exigences fonctionnelles :**
  - Détecteurs de gaz / Alarme
  - Récepteur d'information
  - Contrôleur
- **Scenario de communication :**

Quand le détecteur de gaz détecte une fuite il envoie un message qui prévient qu'il y'a une fuite avec son emplacement au récepteur d'information qui va prévenir le Contrôleur pour que l'alarme se déclenche

- **Scenario de communication selon le protocole CoAP :**

Quand une fuite de gaz est détectée par le détecteur de gaz (cliant1) il envoie un message au récepteur d'informations (routeur) qui va prévenir le contrôleur (serveur) et ce dernier va ordonner à l'alarme (client2) de se déclencher

- **Diagramme de séquence :**



IV.3.2 Pour le protocole MQTT

Le cas étudié Est le même que celui citer avant (scenario de simulation pour le protocole CoAP) sauf que ce système fonctionne avec le protocole MQTT pour cela dans la boulangerie il est installé des capteurs (des détecteurs) de gaz dans les coins susceptibles d'une éventuelles fuite (tuyau, four……) et lorsque une fuite est détecter une alarme se déclenche

- **Exigences fonctionnelles :**
  - Détecteurs de gaz / Détecteur de fumé
  - Contrôleur
  - Alarme
- **Scenario de communication :**

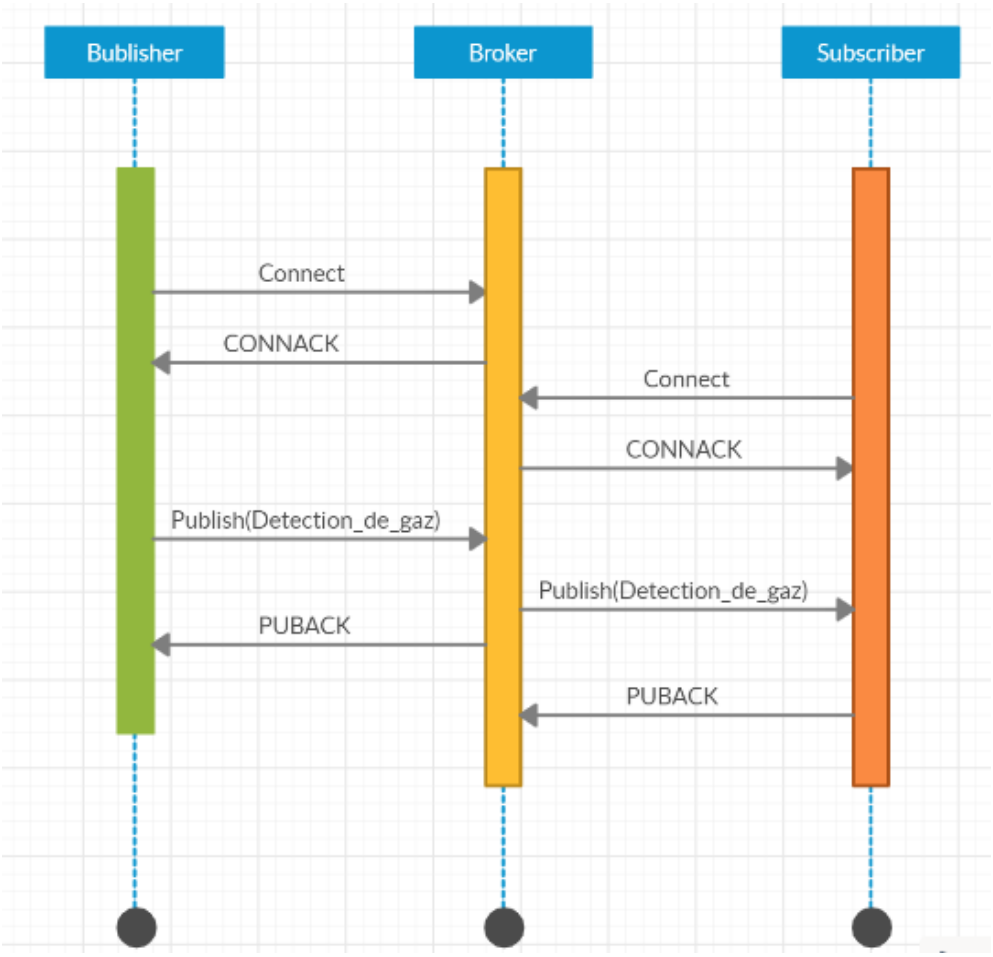
Quand le détecteur de gaz détecte une fuite il envoie un message qui prévient qu'il y'a une fuite avec son emplacement au contrôleur qui va provoquer le déclanchement de l'alarme

- **Scenario de communication selon le protocole MQTT:**

Tout d'abord le détecteur de gaz (publisher) se connecte avec le contrôleur (broker) et le broker accepte la connexion et l'alarme (subscriber) aussi va se connecter avec le broker et s'abonne au topic Detection_de_gaz.

Lors d'une détection de fuite de gaz le détecteur de gaz (publisher) publie dans le topic Detection_de_gaz et le broker envoi à l'alarme (subscriber) comme ça l'alarme se déclenche.

- **Diagramme de séquence :**



IV.4 Environnement de travail et outils de simulation

IV.4.1 Outils matériels

Nous avons utilisé un laptop DELL

- RAM: 8, 00 Go.
- Systém: system exploitation 64 bits.
- Windows 10

IV.4.2 Outils Logiciels

Pour notre simulation nous avons utilisé une machine virtuelle nommée VMware a et le système d'exploitation instant Contiki 3.0 qui met à disposition un simulateur réseau appelé Cooja qui est un émulateur qui permet d'exécuter des programmes sur contiki sans avoir besoin de matériels et pour faire tourner cette machine nous avons utilisé un lecteur des machines virtuelles VMware workstation 15.5

- **VMware Workstation :** VMware Workstation est un outil de virtualisation de poste de travail créé par la société VMware, il peut être utilisé pour mettre en place un environnement de test pour développer de nouveaux logiciels, ou pour tester l'architecture complexe d'un système d'exploitation avant de l'installer réellement sur une machine physique.[29]
- **Wireshark :** Wireshark est un outil open source c'est un analyseur de réseau et de protocole de réseau. Il est anciennement connu sous le nom d'Ethereal, peut être utilisé pour examiner les détails du trafic à divers niveaux allant des informations au niveau de la connexion aux bits qui composent un seul paquet. La capture de paquets peut fournir à un administrateur réseau des informations sur des paquets individuels.

IV.4.3 Les étapes de la simulation

IV.4.3.1 Installation logicielle

- L'installation du système d'exploitation Contiki a été la phase la plus simple.

Installation de mosquitto :

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

IV.4.3.2 Exécution du simulateur Cooja :

- La manière simple d'exécuter Cooja est de l'exécuter dans son propre répertoire figure (4.1)

cd contiki/tools/cooja
ant run

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

Figure 4.1 Exécution de cooja

- Après l'exécution de Cooja, la fenêtre suivante apparaît :

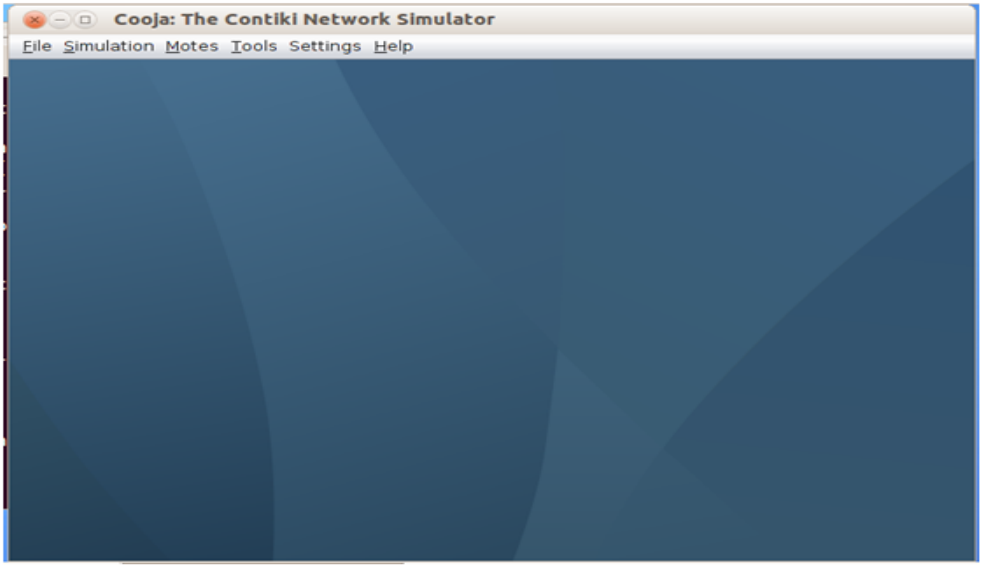


Figure 4.2 Interface de simulateur COOJA

IV.4.3.3 Création d'une nouvelle simulation

Dans le menu il faut choisir : File> New simulation. (Figure 4.3) Apres il faut choisir un nom de la simulation (figure 4.4) ensuite Diverses fenêtres apparaîtront pour la simulation, comme la fenêtre du réseau, fenêtre de contrôle de simulation, sortie de mote et chronologie (Figure 4.5)

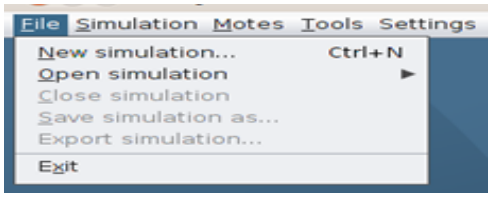


Figure 4.3 Création d'une nouvelle simulation

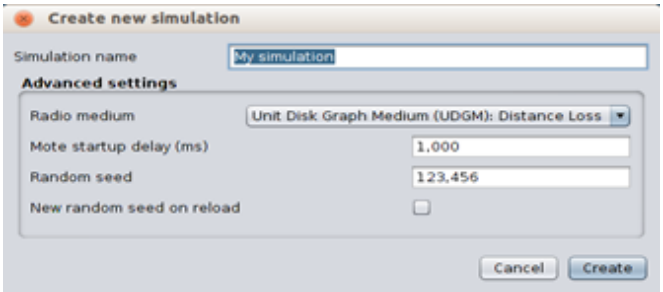


Figure 4.4 Nommer la simulation

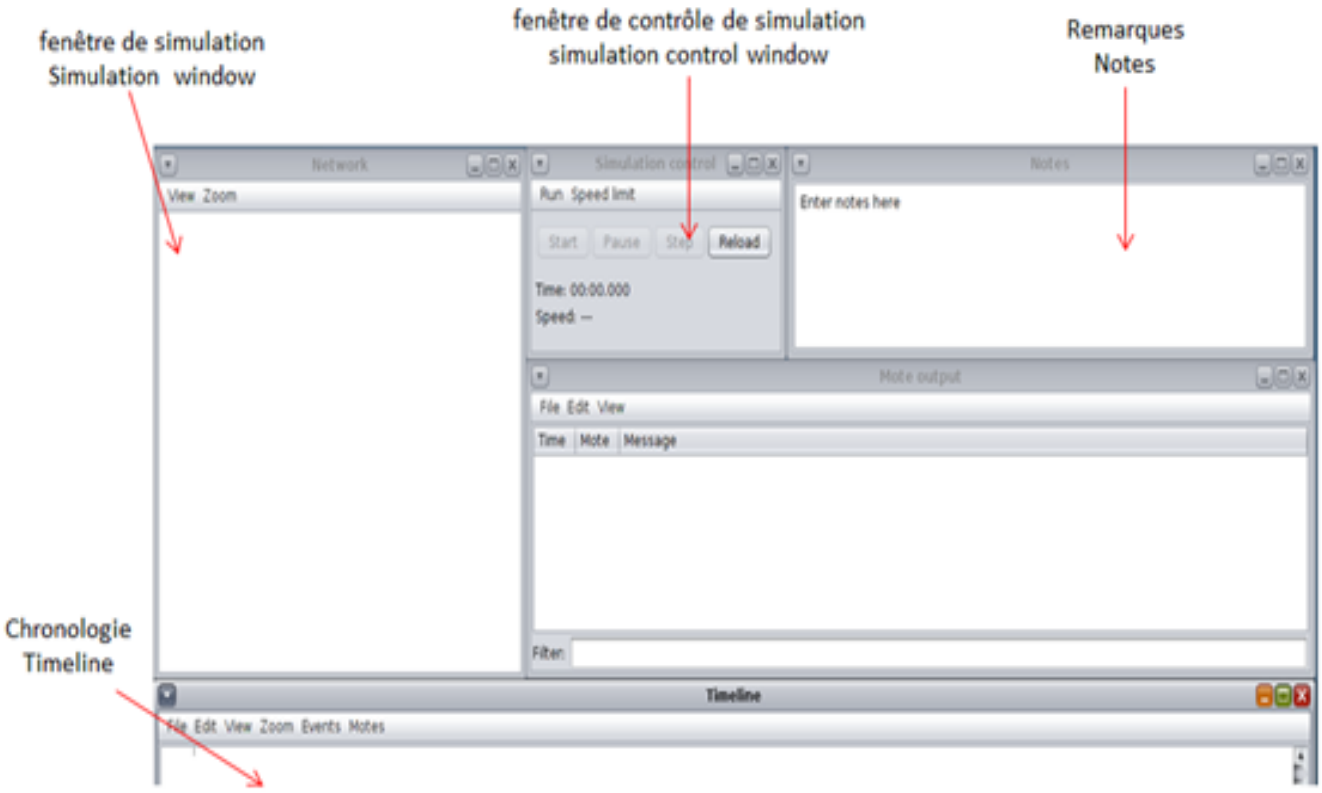


Figure 4.5 Fenêtres du simulateur cooja

IV.4.4 Simulation du scenariodu protocole CoAP

- Une fois que la nouvelle simulation et créé et nommé, il faut créer les motes nous on a choisi des motes de type « Sky Mote» pour le routeur, le serveur et les clients (10) (figure 4.6)

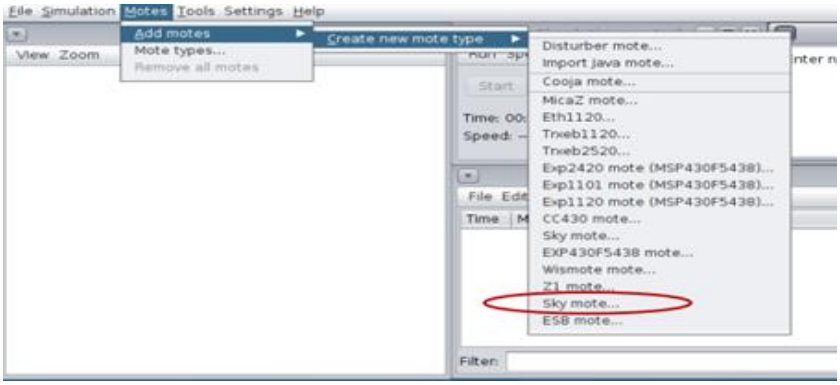


Figure 4.6création des motes

- Après avoir choisi le type de mote une fenêtre apparaît qui permet de donner un ID aux motes et de choisir le fichier .C (en cliquant sur Browse) qui sera compilé comme la compilation d'un programme C et cela va permettre de simuler les motes (Figure 4.7)

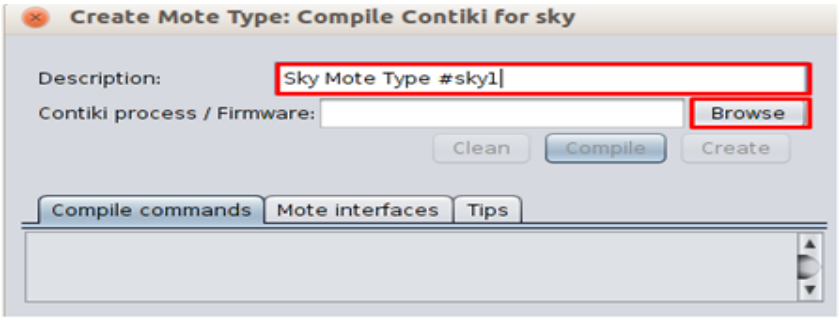


Figure 4.7 Fenêtre de compilation

A. Le router : Nous lui avons donné l'ID « broder router » et on a sélectionné le fichier border-router.c puis on a compilé (en cliquant sur Compile) une foi que la compilation est fini on a choisi de créer notre mote (Create) et une autre fenêtre pour donner le nombre de mote qu'on souhaite créer et leur position est apparu on a choisi 1 (figure 4.8) et enfin on a ajouté notre routeur

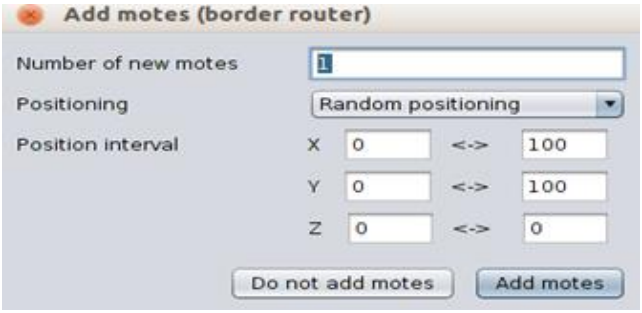


Figure 4.8 Ajout des motes

- **<u>Information du sky mote broder routeur :</u>**

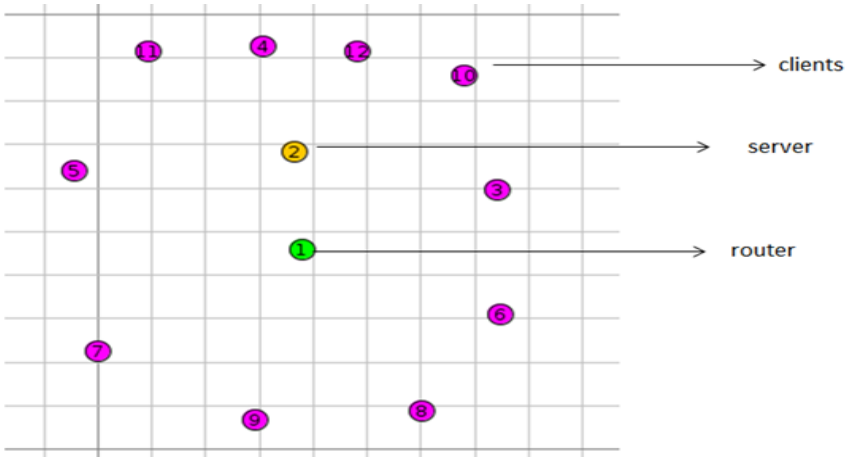
B. Le serveur : même chose que le routeur sauf que on lui a donné l'ID « Server » et on a sélectionné le fichier er-example-server.c

- **<u>Information du sky mote server :</u>**

C. Les clients : On a nommé le sky mote pour les clients « client » et on a choisi le fichier er-example-clients.c et pour ce qui est de nombre de motes on a choisi 10

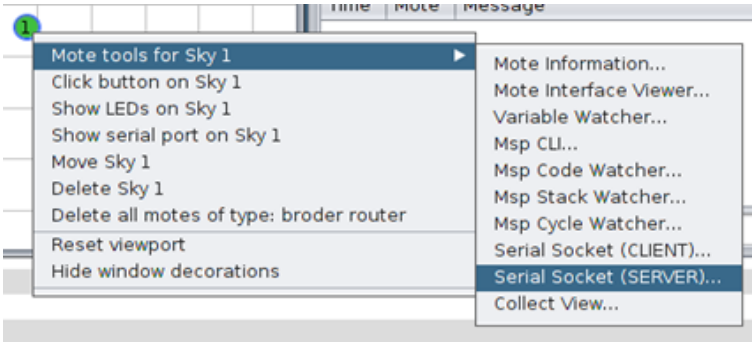
- **<u>Information du sky mote client :</u>**

- Voilà le résultat de création des motes :



- **10 Clients :**9clients qui représentent les Détecteurs de Gaz et 1client qui est l'alarme
- **Server :** Le contrôleur
- **Router :** Récepteur d'informations

Une fois les motes créé il faut allumer le routeur comme suite :



- Après une fenêtre avec l'option de démarrage s'affiche (Figure 4.9)

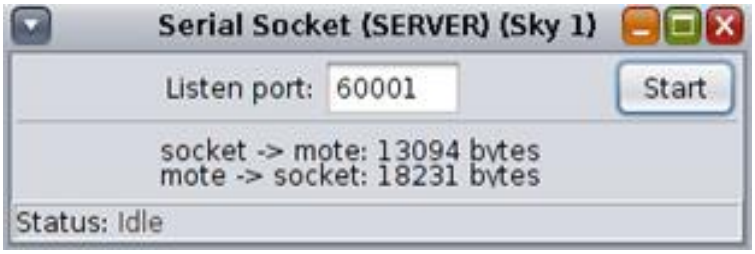


Figure 4.9 fenêtre de serial socket server

- Pour démarrer la simulation et permettre au client de se connecter il faut cliquer sur (start) dans la fenêtre de serial socket server et la fenêtre de contrôle de simulation (Figure 4.10)

Figure 4.10 Commencement de la simulation

- Il faut aussi exécuter la commande "make connect-router-cooja" :

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

IV.4.5 Simulation du protocole MQTT

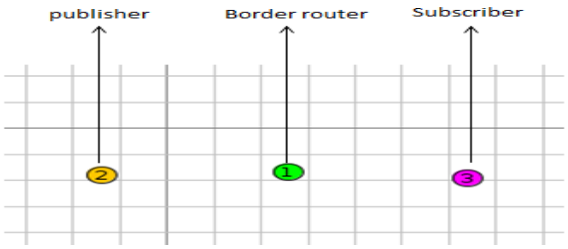
Pour le protocole MQTT on a besoin de 3 motes le routeur le Publisher et le Subscriber

A. Le router : Nous lui avons donné l'ID « border router » et on a sélectionné le fichier a compilé border-router.c

B. Client Publisher : Son ID est « publisher » le fichier a compiler que nous avons sélectionné main_core .c

C. Client Subscriber : Son ID est « subscriber » et pour le fichier a compiler meme chose que le client publisher

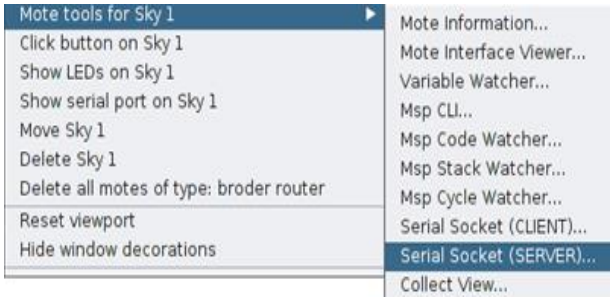
- le résultat de création des motes :



- **Publisher :** qui est le detecteur de gaz

- **Border router :** le controleur
- **Subscriber :** l'alarme

- Une fois les motes créé il faut allumer le routeur :



- Il faut cliquer sur (start) de la fenêtre de serial server, ensuite il faut exécuter la commande make "connect-router-cooja"
- Après il faut ouvrir un nouveau terminal et exécuter la commande "sudo ./broker_mqtts config.mqtt "

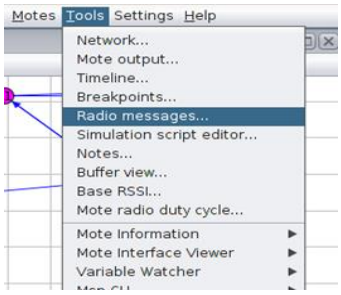
```
user@instant-contiki:~$ cd contiki
user@instant-contiki:~/contiki$ cd mqtt-sn-contiki/
user@instant-contiki:~/contiki/mqtt-sn-contiki$ cd tools
user@instant-contiki:~/contiki/mqtt-sn-contiki/tools$ cd mosquitto.rsmb/
user@instant-contiki:~/contiki/mqtt-sn-contiki/tools/mosquitto.rsmb$ cd rsmb/
user@instant-contiki:~/contiki/mqtt-sn-contiki/tools/mosquitto.rs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

IV.5 Résultats de la simulation du protocole CoAP et MQTT

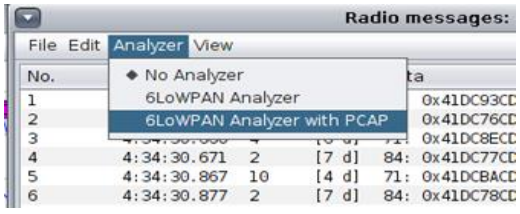
Afin de capturer le trafic d'un protocole dans cooja il possible d'utiliser Wireshark comme suite :

1. **Installer Wireshark**

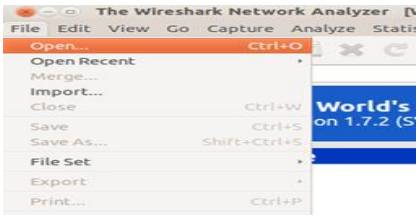
Enregistrer le trafic réseau sous forme de fichier PCAP à l'aide de Cooja Simulator : Sélectionner « Radio message »



2. **Sélectionner :** Analyzer >6LoWPAN Analyzer with PCAP :

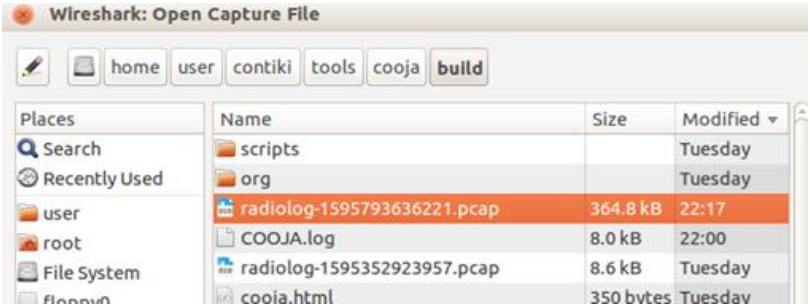


3. Ouvrir, numériser et exporter des fichiers PCAP au format CSV en utilisant le Programme Wireshark : Dans le menu il faut sélectionner : File>Open



4. Selectionner le fichier PCAP

(chemin : /home/user/contiki/tools/cooja/build)



Nous avons utilisé Wireshark pour analyser CoAP

Voici un exemple de l'interface de wireshark :

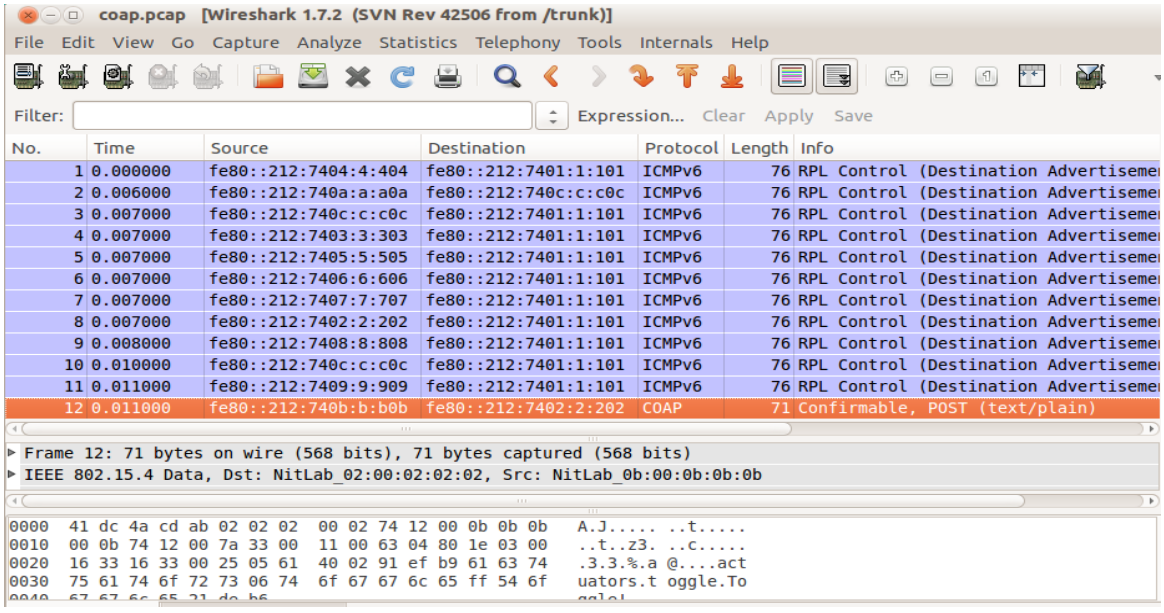


Figure 4.11 Interface Wireshark

La figure 4.12 montre un exemple de récupération d'une transaction capturée par Wireshark.

Différentes couches de réseau de transfert de données dans CoAP sont représentés sur cette figure. Comme on peut le voir sur l'image, le client CoAP démarre la session en envoi d'un 71 octets Messages confirmables (CON) pour acheminer une demande d'obtention vers le serveur CoAP.

Le serveur CoAP a reçu le "POST" au moment "0.011000" et a répondu au client CoAP. (Bleu ligne dans l'image) Le client CoAP a reçu les messages d'accusé de réception CoAP (ACK) avec un 2.05 Type de contenu message (ligne orange sur l'image).

Les frais généraux des couches réseau capturées par WireShark sont affichés au milieu de l'image.

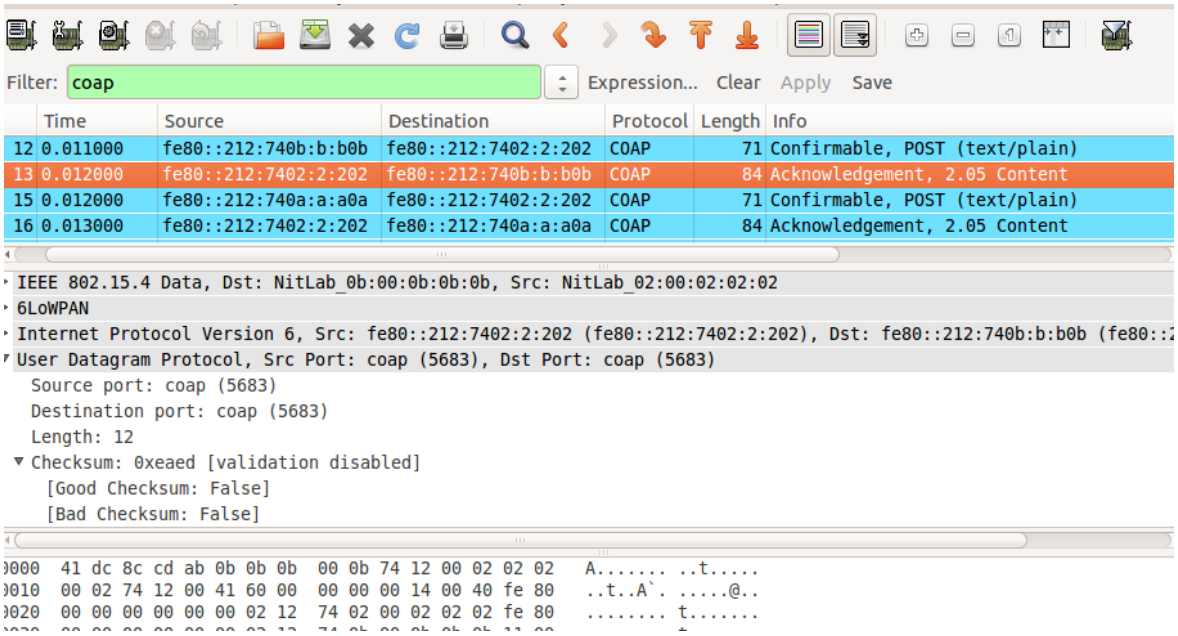


Figure 4.12 Couches réseau dans CoAP

IV.5.1 Comparaison de CoAP et MQTT

IV.5.1.1 Travaux connexes :

Au cours dernières années, certains travaux ont étudié et comparé les performances de l'application protocoles de couche dans les environnements IoT. Parmi les traveaux les plus pertinents :

Travaille	**Résultats**
Dans [56], les auteurs présentent une analyse des besoins en ressources pour les protocoles CoAP et MQTT basé sur des résultats expérimentaux réalisés à l'aide de systèmes Intel X86 avec Ubuntu, libcoap et Mosquitto. Ils sont comparés en fonction de différentes caractéristiques spécifiques du protocole ainsi que l'utilisation de ressources telles que **la bande passante et l'énergie**.	L'analyse des résultats expérimentaux montre que CoAP est le plus efficace en termes de consommation d'énergie et de bande passante. Pour envoyer un message avec une fiabilité pour les deux protocoles, CoAP n'utilise que 140 octets au total (y compris la liaison, le réseau et en-tête de couche de transport). MQTT avec QoS-1 est deuxième avec 162 octets, tandis que MQTT avec QoS-2 nécessite 318 octets au total. **En général, les résultats montrent que CoAP et MQTT consomment une faible bande passante.**
Dans [57], les auteurs comparent les performances de CoAP, MQTT et HTTP REST à l'aide d'un Raspberry Pi 3 et les implémentations aiocoap, Mosquitto et Django des protocoles. Ils comparent **les nombre d'octets consommés et délai pour chacun des protocoles** sous différents réseaux conditions.	Les résultats montrent que **CoAP est le plus efficace en termes de temps et de bande passante pour les plus petits charges utiles**, et ses performances se détériorent à mesure que la taille de la charge utile augmente.

Dans une autre expérience [60] ils ont montré les résultats de **l'influence de la perte de paquets sur le transfert de données**

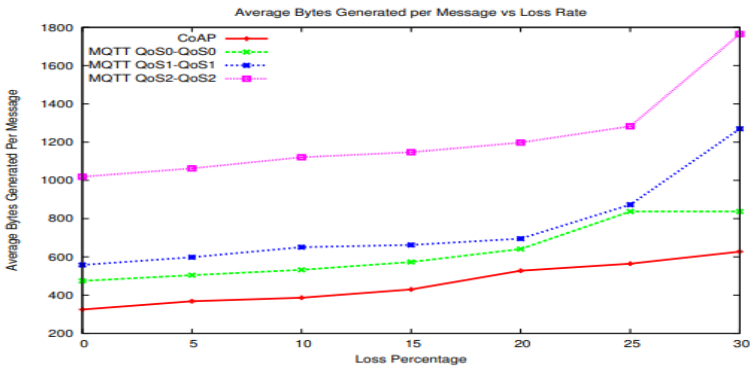


Figure 4.15 Moyenne des données transférées par message par rapport au taux de perte de paquets [60]

Le graphe montre que quand le pourcentage de perte est grand le taux de transfert de message est important.
Les messages QoS 2 occupe plus de bande passante (comme le taux moyen des données transférées augmente tout au long de la simulation) par rapport à QoS 0 et QoS 1.
QoS 0 a le moins de transfert de données.
Pour ce qui est du protocole CoAP quand la taille du message est petite et que le poids de perte est égalà ou moins de 25% il devient plus stable qu'au début donc il génère moins de trafic supplémentaire que MQTT.
La perte de paquet dans CoAP est inferieure que la perte de paquet chez MQTT
Donc le protocole CoAP ne consomme pas beaucoup de bande passante.

IV.5.1.2 Comparaison des performances

Cette section présente, l'évaluation des performances et analyse des protocoles MQTT et CoAP basés sur des valeurs obtenues à partir du logiciel contiki.

Pour notre cas nous allons comparer CoAP et MQTT en terme de **la consommation d'énergie**, **le taux de messages transférées** et cela va nous indiquer l'utilisation de la bande passante. **Influence de la perte de paquets sur le transfert de données**,afin de calculer et suivre ces différents caractéristiques nous avons utilisé quelque méthodes que cooja propose et nous avons aussi utilisé les résultats enregistré sur Wireshark.

➢ **Le taux de message transféré :**

Le client CoAP commence à faire des requêtes en envoyant des méthodes POST. Cependant, la configuration de la connexion dans MQTT est un peu différente de CoAP car plus de messages doivent être échangés en raison du fait que les clients MQTT doivent s'inscrire et s'abonner à un sujet (topic) avant de recevoir un message de publication du broker.
Afin de commencer à recevoir tous les messages de publication liés au topic souscrits, un client MQTT envoie PINGREQ au broker, ce qui est assez similaire à la méthodes POST dans CoAP.
Le tableau 4.1 montre le nombre de message échangé.
Le temps de notre simulation est de 20min
Nous avons choisi de faire la comparaison entre le message Confirmable (POST) du protocole CoAP et le message PINGREQ de MQTT.

Temps	**0**	**5**	**10**	**15**	**20**
Nbr de message CoAp transféré	**0**	**27**	**55**	**82**	**109**
Nbr de message MQTT transféré	**0**	**24**	**49**	**74**	**99**

Tableau 4.1 nombre de messages transféré durant la simulation

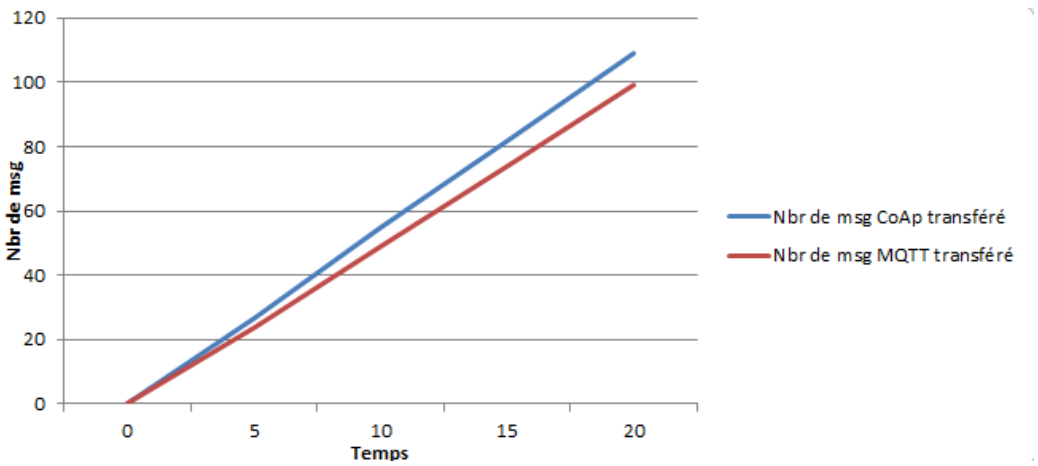


Figure 4.13 Nombre de messages échangés durant la simulation

Comme on peut voir le nombre de messages transféré pour le protocole CoAP est supérieure à celui du nombre de message transféré dans le protocole MQTT cela est dû au fonctionnement du MQTT car au début MQTT échange plus dans le but de s'abonner et enregistrer les Topic,

Une fois que Le client MQTT est abonné aux Topic qui l'intéresse, il passe en «mode veille», ce qui explique la baisse de nombre de message échanger.

- **La consommation d'énergie :**
  - **La consommation d'énergie selon la longueur des messages :**

Apres avoir vu les résultats de taux de messages transféré durant la simulation et vu que le protocole CoAP transfère plus de message on déduit que CoAP consomme plus d'énergie car comme on la explique le protocole MQTT transfère plus de message au début après le Transfer de message baisse et ce qui fait que la consommation d'énergie baisse aussi et il ne faut pas aussi oublie que la longueur du message CoAP sont plus grandes que la longueur du message MQTT tableau 4.2

Pour cette comparaison nous avons choisi les messages que MQTT utilise au début de la simulation qui sont : CONNECT, SUBSCRIBE, PINGREQ, PUBLISH avec le message CoAP Confirmable sous la méthode POST.

	MQTT				CoAP
Type de message	CONNECT	SUBSCRIBE	PINGREQ	PUBLISH	Confirmable
Longueur (bits)	64	54	61	64	71

Tableau 4.2 Taille des messages CoAP et MQTT

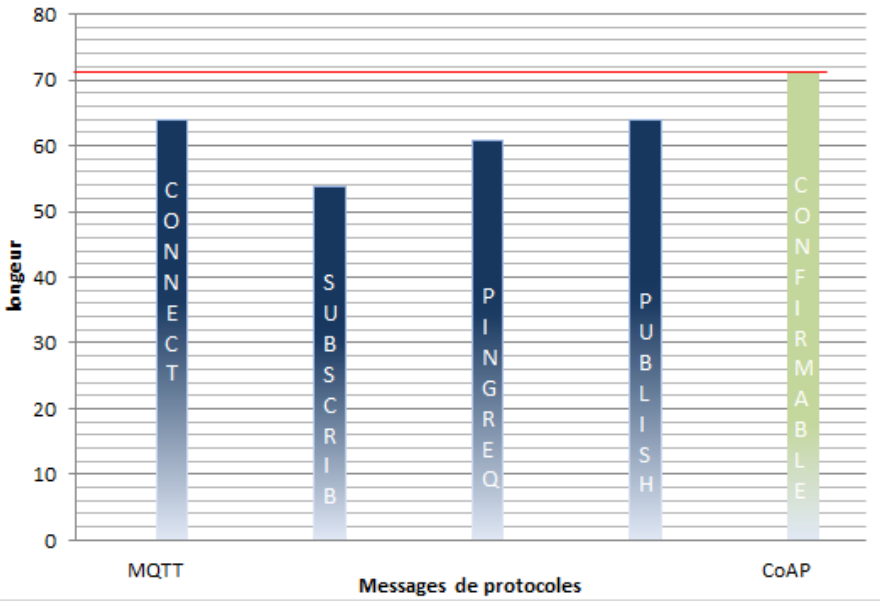


Figure 4.14 Taille des messages CoAP et MQTT

Comme on peut voir les résultats montre que la longueur du message CoAP est supérieure à celle des messages MQTT qui ne dépasse pas le 64 bits

IV.5.1.3 Analyse des performances

Apres avoir analysé les performances des deux protocoles par rapport à :

- la consommation d'énergie ou on a vu que le protocole CoAP consomme plus d'énergie que MQTT du a la taille de ses message transféré.
- On a aussi remarqué que le protocole CoAP transfère plus de message pendant la simulation que le protocole MQTT.

Nous pouvons dire que les deux protocoles sont performants mais leur utilisation dépend des besoins de l'utilisateur et des conditions du réseau.
Pour notre cas les performances du protocole CoAP nous intéressent plus car pour notre scenario, des messages de taille petite nous suffisent. Une autre chose, il ne faut pas oublier que le protocole MQTT nécessite une infrastructure qui implique l'allocation du système du Broker. Cette demande n'est pas pratique lorsque le réseau est déjà déployé. Cependant, la mise en œuvre de CoAP en termes d'infrastructure est plus simple car elle peut fonctionner sans elle.

IV.6 Conclusion

Ce chapitre a été consacré à la simulation des deux protocoles CoAP et MQTT pour qui nous avons proposé un scenario qui fonctionnent selon ces deux dernier.

Pour cette simulation nous avons utilisé l'outil de simulation COOJA qui contient différents méthodes et outils qui nous ont permis d'obtenir a des résultats concernant les performances des protocoles : en terme de la consommation d'énergie, le taux de messages transférées et cela va nous indiquer l'utilisation de la bande passante, et le retard dans les protocoles CoAP et MQTT.

On a fini par comparer ses performances. Les résultats de cette comparaison indiquent que le protocole CoAP, pour notre cas, est plus performant que le protocole MQTT malgré sa grande consommation d'énergie.

Conclusion générale

L'IoT, de nos jours est devenu très fondée, et dans les prochaines années, la plupart des objets de la vie quotidienne seront connectés entre eux et à Internet. En quelques années seulement depuis son apparition, elle fut adopté dans divers secteurs et cela à grâce à son potentiel énorme.

La communication constante entre les objets se fait par de multiples protocoles qui offrent des performances qui permettent à l'IoT d'avoir plus d'impact et d'être plus performant. Parmi les protocoles les plus utilisés pour la communication entre objets dans l IoT, les protocoles CoAP et MQTT qui sont parmi les protocoles les plus connu de l'IoT et c'est pour cela que nous avons voulu évaluer leurs performances et comprendre leur fonctionnement, afin de réaliser une comparaison entre eux.

L'objectif principal de notre travail est d'analyser et montrer l'efficacité de ces deux dernier, cella à l'aide d'une étude par simulation en utilisant le simulateur COOJA, pour l'objectif d'aider les utilisateurs à mieux choisir le protocole qui convient le mieux à leur projets et qui répond à leur besoins.

Nous avons montré les performances de chaque protocole selon la consommation d'énergie, le taux de messages transférées et l'influence de la perte de paquets sur le transfert de données.

Ce travaille nous a permis de :

- Comprendre quelque aspect de l'IoT et les différents domaines de cette dernière.
- Acquérir des connaissances sur la simulation et le fonctionnement du simulateur COOJA.
- Connaitre les différents protocoles de l'IoT et avoir une idée sur les différences qui existent entre eux.
- Mieux comprendre le fonctionnement de MQTT et CoAP.
- Réalisé une comparaison entre les deux protocoles MQTT et CoAP.

Comme résultat de ce travail, nous avons conclu que le protocole CoAP est le plus performant grâce aux résultats de comparaison, mais cela est dans la seule condition qui est la non influence, dans l'application IoT, de la grande consommation d'énergie. On peut dire aussi que le MQTT est le mieux utilisé si la consommation d'énergie est un critère important dans une application IoT.

Comme perspective nous espérons pouvoir vérifier les autres caractéristiques par simulateur et faire une vérification formelle aux deux protocoles et proposer une amélioration pour qu'ils soient plus performants et nous espérons que nous continuerons de cette manière pour l'étude des autres protocoles de l'IoT.

Webographie

[1] https://www.talsom.com/insights/definition-internet-des-objets/(Consulté le 09 mars 2020)

[4] https://www.digora.com/fr/blog/definition-iot-et-strategie-iot(Consulté le09 mars 2020)

[6] https://fr.wikipedia.org/wiki/Mod%C3%A8le_OSI(Consulté le09 mars 2020)

[8] *https://iot.ieee.org/definition.html*(Consulté le09 mars 2020)

[9] *https://www.itu.int/en/ITU-T/gsi/iot/Pages/default.aspx*(Consulté le09 mars 2020)

[10] https://blog.engineering.publicissapient.fr/2018/04/16/internet-des-objets-quels-protocoles-applicatifs-utiliser-1-2/ (Consulté le 09 mars 2020)

[12] https://azure.microsoft.com/fr-fr/overview/internet-of-things-iot/iot-technology-protocols/(Consulté le09 mars 2020)

[14] *https://www.synox.io/4-choses-a-savoir-sur-linternet-des-objets/*(Consulté le09 mars 2020)

[15] https://thenewstack.io/mqtt-protocol-iot/ (Consulté le09 mars 2020)

[17] https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices/ (Consulté le09 mars 2020)

[18] http://www.hivemq.com/blog/mqtt-essentials-part-9-last-will-and-testament (Consulté le09 mars 2020)

[19] http://programmingwithreason.com/article-mqtt-in-depth.html (Consulté le20 mars 2020)

[20]http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html (Consulté le20 mars 2020)

[21] https://openlabpro.com/guide/mqtt-packet-format/ (Consulté le20 mars 2020)

[22] https://www.digi.com/resources/documentation/digidocs/90001525/reference/r_example_mqtt.htm(Consulté le20 mars 2020)

[23] https://www.hivemq.com/blog/mqtt-essentials-part-4-mqtt-publish-subscribe-unsubscribe/ (Consulté le20 mars 2020)

[25] https://fr.wikipedia.org/wiki/CoAP (Consulté le 12 Avril 2020)

[27] https://www-npa.lip6.fr/~fourmaux/res/res4m8c.html(Consulté le 12 Avril 2020)

[28] http://www.nessi2.de/(Consulté le 12 Avril 2020)

[29] https://fr.wikipedia.org/wiki/VMware_Workstation_Pro (Consulté le 12 Avril 2020)

[31] https://fr.wikipedia.org/wiki/Extensible_Messaging_and_Presence_Protocol (Consulté le15 Avril 2020)

[33] https://www.webchoiceonline.com.au/fundamental-characteristics-that-makes-the-internet-of-things-what-it-is/ (Consulté le15 Avril 2020)

[35] http://www.wikiwai.com/2020/02/21/internet-des-objets-architectures-protocoles-et-applications_/(Consulté le15 Avril 2020)

[36] https://www.educba.com/introduction-to-iot/(Consulté le15 Avril 2020)

[39] https://fr.wikipedia.org/wiki/Transport_Layer_Security#:~:text=La%20Transport%20Layer%20Security%20(TLS,r%C3%A9seau%20informatique%2C%20notamment%20par%20Internet.(Consulté le20 mars 2020)

[40] https://www.3cx.fr/webrtc/dtls/(Consulté le20 mars 2020)

[42] https://www.nsnam.org/about/what-is-ns-3/(Consulté le20 mars 2020)

[47] https://www.ideas4allinnovation.com/innovators/here-is-santander-city-brain-the-ideas-platform-launched-by-santanders-city-council-in-collaboration-with-ideas4all/

[48] https://www.postscapes.com/internet-of-things-protocols/(Consulté le 20 mai 2020)

[50] https://www.slideshare.net/butler-iot/butler-project-overview-13603599(Consulté le20 avril 2020)

[52] https://omnetpp.org/intro/(Consulté le10 mai 2020)

[53] https://fr.wikipedia.org/wiki/Simulation_informatique(Consulté le10 mai 2020)

[54] http://www.contiki-os.org/(Consulté le10 mai 2020)

[61] https://www.geospatialworld.net/news/iot-sensors-market-expected-grow-42-08-cagr-2022/(Consulté le 24 septembre 2020)

[62] https://haptik.ai/(Consulté le 24 septembre 2020)

[63] https://www.semanticscholar.org/paper/A-CoAP-gateway-for-smart-homes-Bergmann-Hillmann/0059e65915b93f5c5d3cfb2b4689f2e570c8c2e3(Consulté le 24 septembre 2020)

[66] https://www.lebigdata.fr/mqtt-tout-savoir (Consulté le 24 septembre 2020)
[67] https://www.eclipse.org/community/eclipse_newsletter/2014/february/article2.php?fbclid=IwAR0h_n85YTIc_QsjmlzJfNt6HiBQD0Pd4hiEWG74gQO_qBeTISjfZvBRV3k(Consulté le 24 septembre 2020)

[68] https://www.boson.com/netsim-cisco-network-simulator(Consulté le 24 septembre 2020)

Bibliographie

[2] Yacine Challal. Sécurité de l'Internet des Objets : vers une approche cognitive et systémique. **[en ligne]**Réseaux et télécommunications [cs.NI]. Université de Technologie de Compiègne, 2012. Disponible sur <https://tel.archives-ouvertes.fr/tel-00866052 > (Le 09 mars 2020)

[3]Pierre-Jean Benghozi, Sylvain Bureau, Françoise Massit-Folea. L'Internet des objets. Quels enjeux pour les Européens ?**[en ligne].**Octobre 2008. Disponible sur : <https://hal.archives-ouvertes.fr/hal-00405070> (Le 09 mars 2020)

[5] D.Jain, P.Krishna, V. Saritha. A study on internet of things based applications.**[en ligne].**School of Computing Science and Engineering VIT University, Vellore, TN, India , 2012. Disponible sur <https://arxiv.org/abs/1206.3891>

[7] John A. Stankovic. Wireless sensor networks.**[en ligne].**Juin 2006.Disponible sur:
<https://www.researchgate.net/publication/224333313_Wireless_Sensor_Networks>

[11] Rabeb Saad. Modèle collaboratif pour l'Internet of Things (IoT)**[en ligne].**Mai 2016.Université du Quebec a Chicoutimi. Disponible sur :
<https://constellation.uqac.ca/3983/1/Saad_uqac_0862N_10207.pdf >

[13] MELITI Nedjema. Architecture Basée Agents pour le diagnostic d'un système d'IoT (Internet ofThings). **[en ligne].** Mémoire de Master académique, Architectures Distribuées. Oum-El-Bouaghi : UniversitéLarbi Ben M'hidi Oum-El-Bouaghi, 2017. Disponible sur : <http://bib.univ-oeb.dz:8080/jspui/bitstream/123456789/6894/1/VertionFnalNedjma2017.pdf>(Consulté le09 mars 2020)

[16] MQTT Version 3.1.1. Edited by Andrew Banks and Rahul Gupta. 29 October 2014. OASIS

[24] *JOSUE, Melka. Model Based Testing des applications du protocole MQTT* **[en ligne].** *Mémoire de Master Recherche Informatique. France : Septembre 2016 Disponible sur : <https://fr.slideshare.net/ymelka/model-based-testing-des-applications-du-protocole-mqtt>*

[26] EFORT. COAP (Constrained Application Protocol) : Protocole d'Application pour l'Internet des Objets. **[en ligne].**2015 Disponible sur : <http://www.efort.com/>

[30] IEEE Standards Association et al. P2413-standard for an architectural framework for the internet of things (iot). Institute of Electrical and Electronics Engineers, New York. **[en ligne].**2016. Disponible sur <https://docbox.etsi.org/workshop/2014/201412_m2mworkshop/s04_standards/ieee_p2413_nappey.pdf>

[32] Fatima Zahra Saadaoui, Abderrahim Maizate, Mohammed Ouzzif. État d'art sur les protocoles en temps réel pour l'internet des objets sous le réseau NDN.**[en ligne].** Ecole Supérieure de Technologie de Casablanca (Maroc), Institut Universitaire de Technologie d'Aix Marseille (France), , CASABLANCA, Maroc. Juin 2019 Disponible sur <https://hal.archives-ouvertes.fr/hal-02296848>

[34] GS1. Gs1 and the internet of things, 2017.(Consulté le15 Avril 2020)

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

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

[41] S.Bendel, T.Springer, D.Schuster, A.Schill, , R.Ackermann, &M.Ameling, (2013, March). A service infrastructure for the internet of things based on XMPP. In Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on (pp. 385-388). IEEE.

[43] G.Yoon,J.Choi, H.Park, &H.Choi, Topic naming service for DDS. Janvier 2016, International Conference on Information Networking (ICOIN) Disponible sur <https://ieeexplore.ieee.org/document/7427138>

[44] Hyun-Ji Kim, Ok-Kyoon Ha, Yong-Kee Jun, and Hee-Dong Park . Case study message Races in Data Distribution Service Programs.International Journal of Software Engineering and Its Applications **[en ligne].** 2016. Disponible sur <https://www.researchgate.net/>

[45] S.Feng, J.Cerles, H.Dalmas, T.Do-Khac, et B.Paulin. Sécurité des objets Connectés. **[en ligne].**Institut national des hautes études de la sécurité et de la justice, 2014. Disponible sur <https://www.cigref.fr/archives/entreprises-et-cultures-numeriques/wp/wp-content/uploads/2014/12/rapport-auditeurs-cigref-inhesj-securite-objets-connectes.pdf>

[46] Makkad Asim. A Survey on Application Layer Protocols for Internet of Things (IoT). **[en ligne].**Department of Computer Science and Engineering Institute of Technology, Nirma University Ahmedabad, India Disponible sur <https://www.plm.automation.siemens.com/global/en/webinar/low-code-development/76318>

[49]Sangyoon Oh, Jai-Hoon Kim, and Geoffrey Fox. “Real-time Performance Analysis for Publish/Subscribe Systems”. In: Future Gener. Comput. Syst.26.3 (Mar. 2010), pp. 318–323. ISSN: 0167-739X. DOI: 10.1016/j.future.2009.09.001. URL: http://dx.doi.org/10.1016/j.future.2009.09.001.

[51] OASIS. Advanced message queuing protocol (amqp) version 1.0. Oasis standard, 29 October 2012.

[55] Edosoft Factory, CONTIKI AND TINYOS, August 13, 2012

[56] Bandyopadhyay, S ; Bhattacharyya, A. Lightweight Internet protocols for web enablement of sensors using constrained gateway devices. In Proceedings of the 2013 International Conference on Computing, Networking and Communications (ICNC), San Diego, CA, USA, 28–31 January 2013; pp. 334–340, doi:10.1109/ICCNC.2013.6504105.

[57]Tandale, U.; Momin, B.; Seetharam, D.P. An empirical study of application layer protocols for IoT. In Proceedings of the 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), Chennai, India, 1–2 August 2017; pp. 2447–2451, doi:10.1109/ICECDS.2017.8389890.

[58]A. Velinov et A.Mileva, «Running and Testing Applications for Contiki OS Using Cooja Simulator,» chez International Conference on Information Technology and Development of Education – ITRO 2016, Zrenjanin, Republic of Serbia, Juin 2016.

[59]L. ATZORI, A. IERA et G. MORABITO, «The Internet of Things: A survey,» Computer Networks, pp. 2787-2805, 2010.

[60]Dinesh Thangavel, Xiaoping Ma, Alvin Valera and Hwee-Xian Tan, Colin Keng-Yan TAN. Performance Evaluation of MQTT and CoAP via a Common Middleware. **[en ligne].** Disponible sur <http://hxtan.info/wp-content/uploads/2014-issnip-mqtt-coap.pdf>

[64]Khan R, Khan SU, Zaheer R, Khan S (2012) Future internet: the internet of things architecture, possible applications and key challenges. In: 2012 10th International Conference on Frontiers of Information Technology.

[65]Wu M, Lu T, Ling F, Sun L, Du H (2010) Research on the architecture of Internet of things. In: 2010 3rd International Conference on Advanced Computer Theory and Engineering.