UNIVERSITÉ CHEIKH ANTA DIOP DE DAKAR

ÉCOLE DOCTORALE MATHEMATIQUES ET INFORMATIQUE

ECOLE SUPERIEURE POLYTECHNIQUE

Année : 2020 N°ordre : 157

THESE DE DOCTORAT UNIQUE

Présentée pour obtenir le grade de Docteur de l'Université Cheikh Anta DIOP de Dakar

Mention : Informatique et Télécommunications

Spécialité : Télécommunications

Par :

Kéba GUEYE

Titre : Etude de la convergence des reseaux LTE-A/IoT et ses applications en e-sante, e-formation et e-agriculture

Soutenue le 07/11/2020 devant le jury composé de :

Président	Hamidou	DATHE	Professeur Titulaire	UCAD, Dakar
Rapporteurs	Amadou	S. HASSANE MAIGA	Professeur Titulaire	UGB, Saint-Louis
	Eugène	C. EZIN	Professeur Titulaire	Université d'Abomey-Calavi, Bénin
Examinateurs	Gervais	MENDY	Maître de Conférences (CAMES)	UCAD, Dakar
	Abdourahmane	RAIMY	Maître de Conférences (CAMES)	UCAD, Dakar
Directeur de Thèse	Samuel	OUYA	Maître de Conférences (CAMES)	UCAD, Dakar

Résumé

Dans plusieurs pays africains, des programmes pour atteindre l'émergence accordent une priorité aux secteurs tels que l'agriculture, la santé et la formation qui doivent être développés.

D'une part, plusieurs études ont montré l'apport des appareils ou objets connectés comme accélérateurs de ces secteurs. Ces appareils connectés se caractérisent par leur autonomie et leurs capacités de détection / d'activation, de traitement, de stockage et de mise en réseau. Ils sont capables de recueillir plusieurs types d'informations de prise de décision telles que les conditions météorologiques, le niveau d'humidité d'une zone irriguée, les paramètres biologiques vitaux des utilisateurs, etc.

D'autre part, des chercheurs ont préconisé la relance systématique des séances de travaux pratiques dans les formations en STEM, malgré la massification des effectifs des apprenants, pour que ces derniers puissent réellement développer des compétences à mettre à disposition de l'émergence de leurs pays.

C'est dans ce contexte que ce travail de thèse propose, d'une part un modèle d'espace intelligent aussi bien dans les villes que dans des zones rurales reculées prenant en charge des services de e-santé, e-agriculture et e-formation, et d'autre part une solution de gestion des réseaux complexes des objets connectés utilisés.

En effet, les objets connectés ne pouvant rendre des services adaptés que s'ils échangent des informations pertinentes, nous proposons dans cette thèse l'utilisation des middleware IoT pour atteindre cet objectif.

Nous avons aussi proposé dans cette thèse l'utilisation des réseaux programmés dits SDN pour mieux prendre en charge la complexité de gestion et le nombre important des objets connectés.

Tenant compte des besoins en haut débit et de gestion de latence spécifiques aux objets connectés, de la facilité d'utilisation des plateformes proposées, de la qualité de service et les données à caractère personnel à gérer, nous avons proposé un moyen de fiabilisation du cœur réseau IMS du réseau de transport LTE-A et l'utilisation de WebRTC en vue d'avoir des objets connectés pouvant être gérés ou échangeant des informations à travers le web ; d'où la notion de Web of thing (WoT) abordée dans cette thèse.

Pour rendre beaucoup plus efficace la formation dans le domaine des STEM en général et la télémédecine en particulier, nous avons proposé dans cette thèse une plateforme KMS-IoT permettant aux étudiants en télémédecine de collaborer avec leurs enseignants en vue de diagnostiquer des patients à distance à l'aide des objets connectés.

La plateforme KMS-IoT proposée peut être adaptée et aussi utilisée comme moyen de collaboration à distance entre apprenants et enseignants dans le secteur de l'agriculture de précision notamment en ce qui concerne la prise en compte des diagnostiques des maladies liées aux plantes cultivées ou la gestion de lutte contre les ravageurs des plantes cultivées.

En ce qui concerne la prise en charge des patients atteints de maladies chroniques dans les zones rurales et la télé-pédiatrie, dans les pays en voie de développement, nous avons proposé une plateforme utilisant des API REST, la technologie WebRTC et le concept de WoT pour mettre en place un dispositif de consultation à distance se rapprochant du présentiel. Cette solution pourra contribuer substantiellement à réduire la mortalité infanto-juvénile dans les pays en voie de développement surtout dans les zones rurales dépourvues de spécialistes en médecine.

Mots clés : LTE-A, IoT, WebRTC, WoT, SDN, Middleware, Espace Intelligent, e-santé, e-agriculture, e-formation et STEM.

Abstract

In several African countries, programs to achieve emergence prioritize sectors such as agriculture, health and training that need to be developed.

On the one hand, several studies have shown the contribution of connected devices or objects as accelerators in these sectors. These connected devices are characterized by their autonomy and their detection / activation, processing, storage and networking capabilities. They are able to collect several types of decision-making information such as weather conditions, the humidity level of an irrigated area, the vital biological parameters of users, etc.

On the other hand, researchers have recommended the systematic relaunch of practical work sessions in STEM training, despite the massive number of learners, so that they can really develop skills to make available to the emergence of their country.

It is in this context that this thesis work proposes, on the one hand, a model of intelligent space both in cities and in remote rural areas taking charge of e-health, e-agriculture and e-training, and secondly a solution for managing complex networks of connected objects used.

Indeed, connected objects can only provide suitable services if they exchange relevant information, we propose in this thesis the use of IoT middleware to achieve this objective.

In this thesis, we have also proposed the use of programmed networks called SDNs to better take care of the management complexity and the large number of connected objects.

Taking into account the broadband and latency management requirements specific to connected objects, the ease of use of the platforms offered, the quality of service and the personal data to be managed, we have proposed a means of making the network more reliable. IMS transport and the use of WebRTC in order to have connected objects that can be managed or exchanged information through the web; Hence the concept of Web of thing (WoT) discussed in this thesis.

To make training in STEM in general and telemedicine in particular much more effective, we have proposed in this thesis a KMS-IoT platform allowing telemedicine students to collaborate with their teachers in order to diagnose patients remotely from a distance. using connected objects.

The proposed KMS-IoT platform can be adapted and also used as a means of remote collaboration between learners and teachers in the precision agriculture sector, particularly with

regard to taking into account diagnostics of diseases linked to cultivated plants or the management of pests of cultivated plants.

Regarding the care of patients with chronic diseases in rural areas and tele-pediatrics, in developing countries, we have proposed a platform using REST APIs, WebRTC technology and the concept of WoT to set up a remote consultation device approaching face-to-face. This solution could make a substantial contribution to reducing infant-juvenile mortality in developing countries, especially in rural areas lacking medical specialists.

Mots clés : IoT, WebRTC, WoT, IMS, SDN, Middleware, Espace Intelligent, e-santé, e-agriculture, e-formation et STEM.

Dedicaces

Je dédie cette thèse :

A toute ma famille

A mon défunt père

A ma mère

A mon épouse

A mes enfants

A mes frères et soeurs

A mes oncles et tantes

A tous mes amis

Remerciement

Je tiens à remercier et exprimer ma profonde gratitude au Professeur **Samuel OUYA**, Directeur du Laboratoire d'Infomatique Réesaux et Télécommunications (LIRT) à l'École Supérieure Polytechnique de Dakar, d'avoir accepté de diriger cette thèse avec beaucoup de dextérité et attention, pour ses conseils et pour l'intérêt qu'il a accordé à ce travail.

Je remercie sincèrement le Professeur **DATHE**, Université Cheikh Anta Diop de Dakar, Directeur de l'École Doctorale Mathématiques et Informatique, pour m'avoir accepté dans l'école doctorale pour y mener mes travaux de recherches, et pour avoir accepté de présider ce jury. Merci également pour m'avoir enseigné les bases de l'Analyse Mathématiques en deuxième année de Physique Chimie à la FST.

Je remercie chaleureusement les professeurs **Amadou SEIDOU HASSANE MAIGA** et **Eugène C. EZIN** d'avoir accepté d'être les rapporteurs de ma thèse. Je vous remercie pour avoir consacré du temps à la lecture de mon manuscrit et pour vos remarques constructives qui m'ont permis d'améliorer la qualité de ce manuscrit.

Je souhaite exprimer ma reconnaissance et toute ma gratitude à l'endroit du docteur **Gervais MENDY** et docteur **Abdourahmane RAIMY** d'avoir accepté de participer à mon jury de thèse et les remercies sincèrement.

Mes remerciements vont aussi à l'endroit de tous mes collègues des laboratoires LIRT, RTN et LIMBI.

Tout le personnel enseignant et administratif du Département Génie Informatique de l'Ecole Supérieure Polytechnique de Dakar.

En fin, je remercie chaleureusement tous mes amis et proches qui, au cours de ces trois années de thèse, m'ont toujours soutenu et encouragé, notamment ma mère qui m'a soutenu par ses prières.

Table des matières

Liste des figures

Liste des tableaux

Liste des abréviations

3GPP : Third Generation Partnership Projet

API : Application Programming Interface

COPS : Common Open Policy Service

CSCF : Call State Control Function

DiffServ : Differentiated Services

DTLS : Datagram Transport Layer Security

ETSI : European Telecommunication Standard Institute

GSM : Global System for Mobile Communications

HLR : Home Location Register

HTTP : HyperText Transfer Protocol

IBM : International Business Machines

IETF : Internet Engineering Task Force

IHS : Information Handling Services

IP : Internet Protocol.

IoT : Internet of Things

JSON : JavaScript Object Notation

LTE : Long Term Evolution

M2M : Machine-to-Machine

NGN : Next Generation Network

OASIS : Organization for the Advancement of Structured Information Standards

PSTN : public switched telephone network

QoS : Quality of Service

RBAC : Role-Based Access Control.

REST : Representational State Transfer

RFC : Request for Comments

SDN : Software Defined Networking.

SIP : Session Initiation Protocol

SMS : Short Message Service

SOA : Service-Oriented Architecture

SS : Smart Space

STEM : Science, Technology, Engineering, and Mathematics.

TCP : Transmission Control Protocol

TLS : Transport Layer Security

ToIP : Téléphonie sur IP

UDP : User Datagram Protocol

UMTS : Universal Mobile Telecommunications System

URL : Uniform Resource Locator

VoIP : Voice over Inernet Protocole

XML : Extensible Markup Language

WebRTC : Web Real-Time Communication.

Wi-fi : Wireless Fidelity

WoT : Web of Thing

WTP :Web Transfer Protocol

Introduction Générale

La croissance massive du marché de l'internet des objets (IoT - Internet of Things) et le nombre d'appareils connectés déployés dans le monde entier ne cessent de croitre ces dernières années [1]. Il y aurait actuellement plus de 17, 6 milliards d'appareils connectés et ce chiffre passera à 30,7 milliards en 2020 et à 75,4 milliards en 2025 selon les estimations [2]. De plus, avec l'avancement actuel de l'IoT, IHS estime que chaque utilisateur possède en moyenne environ 3,3 appareils connectés. Ces appareils connectés se caractérisent par leur autonomie et leurs capacités de détection / d'activation, de traitement, de stockage et de mise en réseau. Ils sont capables de recueillir plusieurs types d'informations telles que les conditions météorologiques, le niveau de luminosité d'une pièce, les paramètres biologiques vitaux des utilisateurs (température corporelle, taux de glycémie), etc. Ces informations permettent d'apporter une véritable valeur ajoutée tant du point de vue du consommateur que du producteur de services [3][4].

Les applications IoT couvrent un large éventail de domaines. Elles sont qualifiées intelligentes (smart en anglais) par l'autonomie du service qu'elles fournissent aux utilisateurs finaux. Nous pouvons citer le Smart City, Smart Home, E-Health, Smart Energy, etc. Un aspect important, dans le cas d'une application IoT, est celui lié aux services de santé, en particulier pour les patients vivant en zones rurales et atteints de maladies chroniques telles que l'hypertension artérielle, le diabète etc. Ainsi, de nouvelles solutions sont nécessaires pour répondre aux besoins croissants de soins et d'assistance médicaux, en particulier pour cette catégorie spécifique vivant dans les zones dépourvues de médecin spécialiste.

Les villes intelligentes (smart cities) constituent aujourd'hui un enjeu prioritaire à la fois pour le développement économique des territoires et pour le bien-être et le cadre de vie de leurs habitants. Ainsi, plusieurs études sont menées pour rendre les villes intelligentes où évidemment, les capteurs et actionneurs IoT de la prochaine génération seront déployés. Par ailleurs, la ville intelligente peut être considérée comme une structure de services (santé, énergie, transports, etc.), où chaque service a ses propres défis et doit être analysé en profondeur. Ces capteurs font parti de ce qu'on appelle « Smart Space » (SS) [3][4] qui représente un réseau local parfaitement géré par des objets intelligents (OI). Ces derniers sont accessibles grâce à une passerelle. Celle-ci est directement connectée à un équipement réseau et elle représente l'intermédiaire entre les OI et l'unité de traitement des données. Une ville

intelligente est une ville qui profite à plein de la transformation numérique et des Technologies de l'Information et de la Communication (TIC) pour améliorer les services publics et les rendre plus personnalisés et plus efficients. Une meilleure équité dans l'accès aux services de E-health, E-learning ou E-agriculture est possible en tirant partie des avancées actuelles des TIC, des ressources publiques et des données pouvant être collectées dans toute l'espace intelligente à l'aide des périphériques connectés IoT / WoT.

Or, pour réussir le déploiement de ces villes intelligentes/zones rurales intelligentes, le citoyen doit être replacé au cœur de la stratégie publique. Alors, la connaissance du contexte, en particulier celle de l'utilisateur, peut être utile pour offrir une meilleure qualité de vie. Tous ces services ne peuvent pas être déployés si nous n'avons pas un bon réseau de communication. Mais aussi nous devons connaître les technologies qui vont faciliter l'appropriation de ses services fournis. Cela constitue une chance pour se rattraper dans les domaines du E-Health, E-Learning ou E-Agriculture.

En effet, avec l'apparition de WebRTC qui est une technologie récente de communication permettant d'établir des échanges multimédias conversationnels directement entre navigateurs en temps réel. Sa richesse et sa versatilité laissent présager des opportunités inédites en termes de services de communication innovants. En introduisant une convergence native des services de communication synchrone / asynchrone, fixe / mobile, voix / donnée, WebRTC fait du web le support naturel et définitif de tout service de communication professionnel ou du grand public. Donc une nouvelle manière d'améliorer le flux multimédia avec les informations contextuelles de l'utilisateur devient possible. Le client WebRTC peut communiquer directement avec les objets intelligents s'ils disposent de suffisamment de fonctionnalités pour établir un canal (HTTP ou autres), ou avec la passerelle qui servira d'intermédiaire entre les utilisateurs et les périphériques sous contrainte.

De plus, le Web of Things (WoT) peut être considéré comme une spécification de l'IoT. Il résume principalement toute la complexité de la partie connectivité de l'IoT et accorde plus d'importance aux applications et aux services. La force du WoT réside dans le fait qu'il fournit une couche d'applications standards et interopérables basées sur le Web. L'idée principale est que tous les objets intelligents peuvent communiquer à l'aide d'un langage Web via une interface standard d'échange de données.

Par conséquent, ce type particulier d'interactions nécessite la résolution de problèmes supplémentaires liés à ces objets intelligents environnants et les entités (Middleware ou web

service) permettant de faire l'échange de données issues des périphériques IoT. Certaines de ces questions sont principalement liées à :

- La collecte des données, généralement issues d'un environnement hétérogène d'objets intelligents, communicants à l'aide de différents protocoles de communication, peut être présente dans différentes architectures IoT qui ne sont généralement pas interopérables ;
- La découverte de ces objets et leur enregistrement ;
- La qualité de service, la sécurité et la confidentialité, puisque ces capteurs collectent des informations directement liées à l'utilisateur, ce qui a donc un impact direct sur la vie privée de celle-ci.
- La facilitation d'accès aux services offerts par l'IoT pour les populations vivant en zone rurale

Ces problèmes deviennent plus complexes lorsque nous traitons avec plusieurs objets intelligents situés dans différents espaces intelligents. La gestion séparée de chaque espace intelligent peut être une tâche ardue pour l'administrateur, en particulier en ce qui concerne la partie qualité de service. Dans une infrastructure intelligente unifiée comportant plusieurs espaces intelligents, les stratégies de QoS et les mécanismes d'automatisation des couches d'accès et de transport doivent être parfaitement gérés afin d'éviter toute collision entre les règles et faille de sécurité.

Ensuite, en contrôlant l'accès, de sorte que seuls les utilisateurs légitimes soient autorisés à accéder aux ressources correspondantes. Enfin, en chiffrant l'interaction avec le SS afin de garantir la confidentialité et l'intégrité des données. Celles collectées à partir du SS peuvent être fournies à l'aide de plusieurs types de protocoles en fonction du service ciblé.

Les premiers types de services sont ceux qui ne nécessitent pas de transfert de données en temps réel. Ils utilisent principalement les protocoles Web traditionnels pour transporter les flux de données de manière sécurisée, tels que HTTPS, WebSocket Secure etc. Ces types de services peuvent être dirigés soit vers une entité (par exemple le navigateur Web d'un administrateur), vers des serveurs et des clouds pour le traitement et la gestion, ou vers des bases de données pour le stockage.

Les secondes nécessitent un transfert instantané et en temps réel des données. Nous proposons une solution originale à ce problème en utilisant les canaux de données fournis par WebRTC. Il ouvre la porte à toute une gamme de nouveaux services couplant les flux WoT aux

flux multimédia de WebRTC mais aussi celle utilisant le protocole SIP. Un tel couplage prend tout son sens dans la vue centrée sur l'utilisateur : les flux WoT peuvent représenter des informations contextuelles associées à l'espace intelligent de l'utilisateur, tandis que les multimédia concernent ses services de conversation. Il est principalement destiné à une entité, mais sans s'y limiter. Les défis consistent à concevoir une architecture permettant de déployer et d'exploiter de tels services de manière sécurisée et respectueuse de la vie privée.

L'objectif de cette thèse est, d'une part de proposer, dans le contexte d'une ville intelligente/zone rurale intelligente, un modèle d'espace intelligent, de déploiement et d'interaction entre les acteurs des structures de services, E-Health, E-Learning et E-Agriculture en s'appuyant sur les TIC. Et d'autre part contribuer à la collaboration entre Infirmiers, médecins généralistes et spécialistes par l'intermédiaire d'un middleware IoT dans le but de faciliter l'accès à des soins de qualité aux populations vivant en zone rurale et de réduire le coût de la prise en charge en se basant sur la technologie WebRTC, WoT, SDN et IMS.

Contributions

Les contributions proposées dans cette thèse portent globalement sur sept publications scientifiques dont deux IEEE, quatre springer et une IJACSA. Ces contributions se déclinent comme suit :

1. **Proposition d'une solution d'accès universel aux soins dans les zones rurales : Cas du Sénégal (IEEE)**

Pour satisfaire l'objectif d'accès universel à des soins de qualité et à moindre cout dans les zones rurales, une solution de monitoring portatif est proposée dans cette contribution. Le système permet à l'infirmier de choisir un spécialiste si une valeur anormale des paramètres physiologiques des maladies chroniques à savoir l'hypertension artérielle, le diabète mais aussi de permettre aux spécialistes d'avoir une idée sur la température corporelle et l'électrocardiogramme du patient, est captée par les capteurs biométriques. Cela aide à prendre les mesures appropriées instantané, et d'anticiper sur les éventuels risques de maladies qui peuvent subvenir à l'avenir. Un tel système peut aider à réduire les factures d'hôpitaux découlant de l'admission du patient à l'hôpital et l'accès à de soins de qualités tout en restant dans sa localité.

2. **Proposition d'un système de soins de santé piloté par l'IoT et KMS pour la surveillance à distance des patients dans les zones rurales : Cas de la Pédiatrie (IEEE ; Best Paper)**

Nous proposons ensuite une plateforme utilisant des API REST, la technologie WebRTC et le concept de WoT pour mettre en place un dispositif de consultation à distance se rapprochant du présentiel. Cette solution pourra contribuer substantiellement à réduire la mortalité infanto-juvénile dans les pays en voie de développement surtout dans les zones rurales dépourvues de spécialistes en médecine. Kurento Media Serveur (KMS) permet de créer des applications de traitement de média basées sur le concept de pipelines. Le Web of Things (WoT), considéré comme un sous-ensemble de l'Internet des objets (IoT), se concentre sur les normes et les frameworks logiciels tels que REST, HTTP et URI pour créer des applications et des services qui combinent et interagissent avec une variété de périphériques réseau.

3. **Modèle de contrôle d'accès dynamique basé sur les délégations géo-temporelles dans les systèmes d'e-santé d'objets connectés. (Springer)**

Notre troisième classe de contribution porte sur la proposition d'un modèle basé sur la délégation dynamique des rôles, en mettant l'accent sur le travail collaboratif et la protection de la vie privée des patients. Ce modèle est une redéfinition du modèle ORBAC prenant en compte la notion d'attributs utilisateurs. Nous utilisons la logique du premier ordre et la logique non monotone **T-JClassic$\delta\varepsilon$** pour effectuer une interprétation axiomatique du modèle. Nous implémentons le modèle avec les technologies WebRTC, Node.js et Kurento Media Server pour faciliter la communication en temps réel entre les utilisateurs et la Raspberry Pi pour collecter les informations biométriques reçues des capteurs.

4. **Contribution à l'amélioration de la base de présence des serveurs VoIP pour l'envoi et la réception de messages (Springer)**

Nous proposons également une quatrième contribution qui s'est accentué sur le protocole de Signalisation SIP afin de contribuer à l'amélioration sur l'envoie et la réception de message sur les services apportés par les serveurs VoIP. La place du protocole SIP est importante dans un système de communication en temps réel plus précisément dans les middleware IoT intégrant IMS. La configuration actuelle des serveurs VoIP nous a permis de constater que si un utilisateur connecté par wifi au serveur VoIP s'est déconnecté involontairement du réseau sans pour autant déconnecter son client SIP du serveur, que ce dernier n'a pas pu l'enlevé de sa base de présence, où il stocke tous les utilisateurs connectés, et du coût le message qu'on lui

envoi n'est pas stocké et non plus le destinataire ne le reçoit pas. Pour pallier à cela, nous couplons l'intelligence de Freeswitch utilisée comme serveur VoIP couplé à un Serveur de détection de présence et une base de données Mysql. Cette plateforme permet récupérer tous les messages à destinataire non connecté et les stockés dans une base de données pour ensuite attendre leur reconnexion afin de leur envoyer les messages qui les concernent.

5. Contribution à la mise en place d'une plateforme de travaux pratiques à distance pour les STEM : Cas de la médecine (ICL)

Le cinquième article vise à contribuer à la mise en place d'une plateforme KMS-IoT de travaux pratiques dans l'enseignement à distance de la médecine. En effet, les étudiants sont géographiquement dispersés et ne disposent pas de laboratoires virtuels pour les Travaux Pratiques. La solution proposée permet au professeur d'initier un TP de consultation médicale en direct via la plateforme KMS-IoT. L'objectif du TP est d'apprendre aux étudiants comment se déroule un examen clinique. Pour ce faire, le professeur utilise les appareils médicaux connectés (stéthoscope, thermomètre, électrocardiogramme, tensiomètre, etc.) pour collecter les constantes d'un étudiant pris comme patient. Ces données sont directement transmises à l'application puis partagées en temps réel avec les autres étudiants via la plateforme. A l'aide d'un ordinateur, d'une tablette ou d'un smartphone connecté, n'importe quel étudiant peut visualiser les constantes et suivre en temps réel les commentaires/explications de l'enseignant. Chaque étudiant peut également interagir en posant des questions.

6. Contribution à la mise en place d'une plateforme de travail pratique à distance pour les STEM : le cas de l'agriculture (Springer)

L'enseignement à distance dans le domaine de l'agriculture, la biologie, etc. est confronté à de multiples difficultés techniques et logistiques. En effet, les visites de terrains représentent un complément indispensable du cours théorique dispensé aux étudiants. Il arrive que les zones abritant la biodiversité recherchée soient instables et potentiellement dangereuses pour les non-résidents. La solution proposée permet aux étudiants d'une institution de formation supérieure d'agriculture de réaliser une excursion pédagogique visant à étudier la biodiversité d'une localité. Pour ce faire, un groupe résident dans la région est sélectionné pour la prise des mesures (humidité du sol, composition chimique, température ambiante, etc.) sous la supervision d'un enseignant. Les données collectées sont directement transmises à l'application puis partagées en temps réel avec les autres étudiants via la plateforme. A l'aide d'un ordinateur, d'une tablette ou d'un smartphone

connecté, n'importe quel étudiant peut visualiser les mesures et suivre en temps réel les commentaires/explications de l'enseignant. Chaque étudiant peut également interagir en posant des questions.

7. **Proposition d'une solution d'automatisation des entités d'une plateforme IMS-KMS-IoT basée sur SDN (IJACSA)**

Enfin, nous avons proposé l'utilisation des réseaux programmés dits SDN comme réseau de transport pour mieux prendre en charge la complexité de gestion et le nombre important des objets connectés. Nous faisons un pas de plus vers l'intégration du SDN dans une plate-forme IoT-IMS en introduisant l'automatisation des éléments de base, afin de répondre aux défis de l'augmentation du trafic et de la complexité du réseau. En sus, nous proposons également l'automatisation de la couche contrôle (plateforme IMS/KMS) et de la couche transport de l'architecture fonctionnelle.

Structure du manuscrite

Ce manuscrit est divisé en cinq chapitres structurés comme suit :

Le premier chapitre traite de l'état de l'art des solutions Middleware LTE/IoT. Nous y analysons, particulièrement les limites des Middlewares à prendre en charge et les besoins en QoS des applications E-Health, E-learning et E-agriculture.

Le deuxième chapitre est consacré au choix des outils et protocoles susceptibles d'être utilisés pour concevoir une plateforme de mutualisation en tenant compte des normes et des bonnes pratiques en matière de Middleware.

Le troisième chapitre traite des middlewares dans le domaine de l'internet des objets. Nous y traitons les concepts et prérequis des middlewares IoT afin de situer leur contexte d'utilisation. Ce chapitre présente aussi les principaux modèles middlewares IoT et identifie ceux d'entre eux qui sont les plus utilisés dans le domaine des e-santé, e-learning et e-agriculture. En somme, ce chapitre va nous servir de guide, pour la conception de stratégies de mutualisation des ressources.

Le quatrième chapitre propose l'utilisation de webRTC en WoT pour bénificier de ses capacités de communication en temps réel de manière à avoir une architecture nouvelle et innovante. Le but principal de cette contribution étant de fournir des services de communication en temps réel enrichis de données contextuelles recueillies à partir des environnements des

utilisateurs. Celles-ci peuvent être collectées à partir de diverses sources, telles que les capteurs IoT.

Etant donné que dans une ville intelligente, il peut y avoir un nombre important de Smart Spaces devant échanger des informations, nous proposons dans le chapitre 5 l'utilisation des réseaux programmés (SDN) pour une bonne gestion et maitrise de ceux-ci. De manière précise nous y avons proposé un mécanisme de gestion de qualité de services et d'automatisation de couches de transport et de contrôle.

1 Etat de l′art sur l’internet des objets

1.1 Introduction

De par la largeur de son spectre et les attendus qu’il suscite, l’IoT fait actuellement l’objet de nombreuses études de recherche, relevant tant de ses applications que de la reconsidération de problématiques initialement abordées dans l’Internet “classique”. Dans ce chapitre, nous faisons un état de l’art sur l’internet des objets (IoT).

D’abord, nous présentons le paradigme de l’internet des objets relatifs à ses applications métiers ainsi que les visions architecturales et chalenge. Ensuite, nous dressons un état de l’art des solutions envisagées au niveau Middleware pour l’IoT. Nous fournissons en particulier une analyse des capacités et des limites de ces Middlewares face aux besoins en QoS pour les applications e-santé, e-formation et e-agriculture. Enfin, nous définissons le concept de WoT notamment la découverte des services et nous faisons la différence entre WoT et IoT.

1.2 Paradigme de l’internet des objets

Le qualificatif d’Internet des Objets (IoT) est apparu pour la première fois en 1999. Ce paradigme est considéré comme étant la prochaine génération de l’Internet [5][6].

Plusieurs définitions ont été données à l’IoT. Gartner [7] le définit comme étant ‘’le réseau d'objets physiques qui contiennent des technologies intégrées pour communiquer et détecter ou interagir avec leurs états internes ou l'environnement externe”. L’IETF considère l’IoT comme étant une extension des technologies de l’actuel Internet et affirme qu’un véritable IoT exige que les objets soient capables d’utiliser les protocoles de l’Internet [8]. Une définition plus détaillée a été donnée par l’IERC (European Research Cluster on the Internet of Things) [9] dans laquelle l’IoT est “une infrastructure de réseau globale dynamique avec des capacités d'auto-configuration, où les objets physiques et virtuels ont des identités, des attributs physiques et des personnalités virtuelles, et utilisent des interfaces intelligentes pour se connecter entre elles et au réseau de données”. Ce paradigme permet de couvrir un large éventail d'applications dans plusieurs domaines tels que la santé, la domotique, les services publics, les transports, ou encore les usines du futur.

Dans cette section, nous illustrons trois applications métier dédiées à l’IoT, l’objectif étant d’illustrer la valeur ajoutée qu’apporte l’IoT aux différentes parties prenantes. Nous

présentons enfin les différentes visions architecturales pour l'IoT ainsi que ses principaux challenges.

1.2.1 Applications métier de l'IoT

Une application métier dédiée à l'IoT (application IoT) se caractérise par l'autonomie ajoutée à une activité métier, telle que la surveillance de patients à distance et l'intervention en cas d'urgence, à travers l'utilisation d'équipements de toutes sortes accessibles via l'Internet.

L'autonomie qu'offre ces applications est destinée à permettre, par exemple, des économies de temps ou à réduire les risques d'erreurs, en minimisant autant que possible l'intervention humaine. Ces applications sont basées sur un ensemble d'échanges réalisés de manière essentiellement autonome entre les équipements, les entités intermédiaires et au final les humains via des moyens de communication, de nouvelles capacités de traitement de données offertes par le cloud computing, et des outils de gestion et d'analyse de ces données.

Les applications IoT couvrent un large éventail de domaines. Elles sont qualifiées d'intelligentes (smart en anglais) par l'autonomie du service qu'elles fournissent aux utilisateurs finaux. Nous parlons à présent de Smart City, Smart Home, e-Health, Smart Energy, etc. Dans cette section, nous étudions trois catégories d'applications IoT, relevant respectivement du Smart Metering, du e-Health et du Smart Factory, dans le but de montrer la valeur ajoutée de l'utilisation de l'IoT du point de vue du consommateur et du fournisseur de services.

1.2.1.1 Smart Metering

L'économie d'énergie est une priorité mondiale pour la préservation de notre planète. Le contrôle de la consommation énergétique permet d'aller vers cet objectif. L'une des applications clés du Smart Grid est la mesure intelligente. Elle est basée sur des compteurs dits intelligents offrant une gamme de services tels que la mesure et l'enregistrement de la consommation (électricité, gaz, eau, etc.), le relevé de la consommation locale ou distant, la fixation du seuil de consommation maximale et l'extinction à distance de l'électricité par le consommateur, ainsi que l'échange d'informations sur la consommation avec les services publics.

Les compteurs intelligents sont considérés comme les successeurs des compteurs d'électricité mécaniques classiques dits muets. La différence est que, pour les compteurs intelligents, il n'y a pas besoin d'intervention humaine (Figure 1.1). Toutes les fonctionnalités se font automatiquement et à distance.

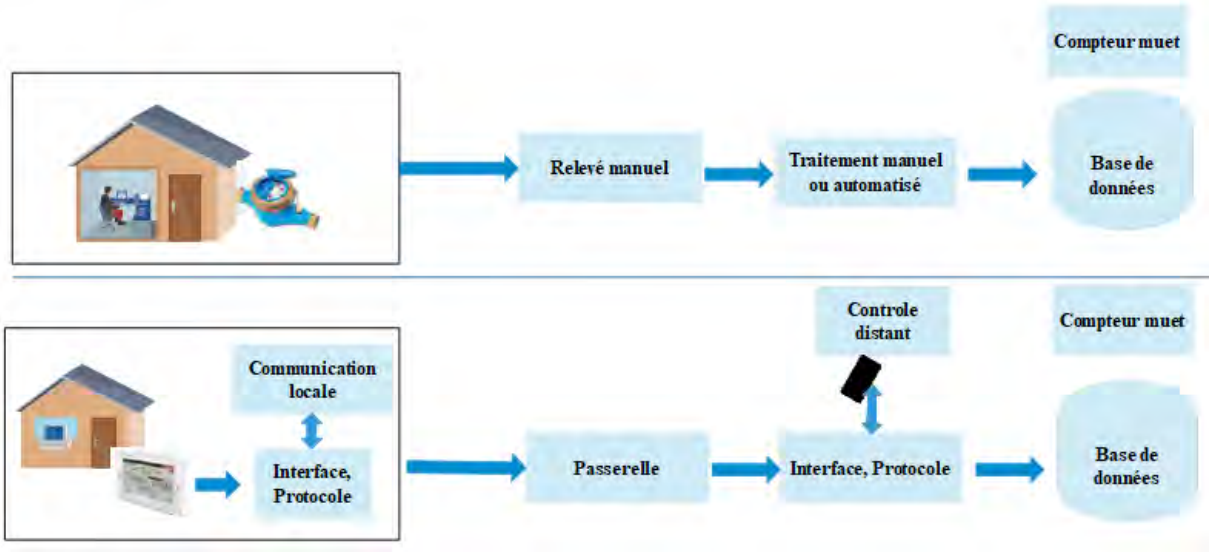

Figure 0.1: Différence entre le compteur traditionnel et intelligent [10]

Au regard des avantages qu'offre une telle application, le consommateur et le fournisseur de services sont bénéficiaires. Le consommateur de service (le client final) a l'opportunité de réaliser des économies d'argent en ayant différentes vues de sa consommation, par exemple, la consommation en temps réel et son équivalent en fcfa ou autres devises, l'historique de la consommation et la visualisation des graphiques connexes, le coût quotidien, mensuel ou annuel, ainsi que sa capacité à contrôler à distance l'utilisation de ses ressources. Du point de vue du fournisseur de services (l'opérateur électrique), les coûts d'acquisition de données sont minimisés. Les agents, dont le rôle était auparavant dédié au déplacement pour l'acquisition manuelle de la consommation, peuvent maintenant être affectés à d'autres tâches. En outre, le fournisseur de services peut déterminer le comportement des utilisateurs liés à leur consommation énergétique pour une meilleure gestion des ressources. Notons que les questions liées à la protection de la vie privée sont sujettes à de multiples études qui ne sont introduites dans ce manuscrit.

1.2.1.2 E-Health

Le monde de la médecine (ou en général la santé) est en constante progression. Il commence à intégrer les méthodes et les ressources des technologies de l'information et de la communication. L'avènement des technologies de l'IoT permet d'envisager un saut substantiel. L'e-Health est l'un des sous-domaines les plus prometteurs de la télémédecine basée sur l'IoT. La Commission européenne le définit comme l'utilisation de technologies d'information et de communication modernes pour répondre aux besoins des citoyens, des patients, des professionnels de la santé, des fournisseurs de soins de santé, ainsi que des décideurs [11]. Les

applications IoT pour l'e-Health permettent la surveillance à distance des informations sur la santé et la condition physique des patients, le déclenchement d'alarmes dans des conditions critiques et, dans certains cas, la maîtrise à distance de certains traitements, ou paramètres médicaux [12].

Dans le passé, il n'y avait pas de surveillance à distance. Le patient devait être présent à l'hôpital, et dépendait des médecins et des équipements hospitaliers pour suivre et enregistrer les informations sur sa santé. Maintenant, les soins aux patients sont améliorés avec la surveillance à distance (Figure 1.2). La personne surveillée utilise des capteurs physiologiques portables formant un réseau BAN (Body Area Network). Ces capteurs recueillent et enregistrent différents signes vitaux tels que la fréquence cardiaque, la température corporelle, la pression artérielle, le taux respiratoire, les chutes et d'autres informations sur le patient. Les données collectées sont envoyées à un agrégateur (par exemple, le téléphone cellulaire du patient) qui est responsable de la transmission d'informations au centre de contrôle [13]. Grâce à ce système, les agents de santé peuvent exécuter des plans prédéfinis et intervenir lorsque des conditions critiques sont détectées en fonction des alarmes déclenchées.

Figure 0.2: Exemple d'architecture de télésurveillance de patient

De cette façon, plusieurs avantages peuvent être énoncés. Du point de vue du consommateur (ici, le patient), il y aura une réduction du temps et des risques. Le patient peut rester et guérir à la maison tout en bénéficiant d'une supervision et d'une assistance en temps réel. Pour le fournisseur de services (ici, le centre hospitalier), il y aura une réduction considérable des coûts grâce à la diminution des services nécessaires. Les hôpitaux seront en outre en mesure de traiter plus de patients sans avoir besoin de plus de lits et de personnels.

1.2.1.3 Smart Factory

Le temps est une clé importante dans le contexte industriel. Tout temps d'arrêt signifie perte de revenus. Les usines recherchent toujours l'amélioration de l'efficacité de la production grâce à la réduction de coûts et de temps. L'intervention des technologies de l'IoT dans le

processus de fabrication dans les usines peut apporter une vraie valeur ajoutée ; nous parlons ainsi d'usines intelligentes (ou smart factories). Dans [14], l'usine intelligente se réfère à la connexion des machines, des dispositifs et de la logistique, et au final des humains pour effectuer la coordination nécessaire de manière omniprésente et ponctuelle. L'usine sera "intelligente" dans différents processus. Elle pourra suivre les produits qui sont dotés d'étiquettes RFID sur la chaîne d'approvisionnement mondiale. Elle sera capable d'obtenir la charge et l'état de chaque machine, et de détecter la panne de la machine et la changer rapidement et pourquoi pas d'une manière autonome sans aucune intervention humaine. Le coût et la qualité de production seront optimisés en utilisant des capteurs intelligents et des actionneurs permettant d'avoir plus de souplesse, une adaptation rapide aux exigences changeantes et une gestion plus efficace des ressources.

Du point de vue consommateur de services (le client), les bénéfices induits par l'usine intelligente se traduisent par la possibilité de suivre l'état d'avancement de sa commande tout au long du processus de fabrication et de livraison, et aussi la possibilité de personnaliser sa commande qui sera prise en compte directement par le fabricant de la machine. Pour le producteur (usine), le coût de gestion et d'intervention en cas de panne sera plus bas grâce à la rapidité de détection de la panne. Il y aura aussi une optimisation du processus de fabrication grâce au suivi en temps réel et distant de l'état du produit, ainsi que la réduction du temps de fabrication en prenant rapidement en compte les changements d'exigences ou de processus.

1.2.2 Visions architecturales et challenges de l'IoT

Plusieurs visions architecturales ont été proposées pour l'IoT. L'architecture basique est constituée de trois niveaux : le niveau Application incluant les services métiers associés, le niveau Réseau qui inclut les différents types de réseaux, et enfin le niveau Équipement comportant l'ensemble des objets impliqués dans la capture d'événements et le contrôle depuis l'environnement (capteurs, actionneurs, etc.).

Dans [15] l'architecture proposée reprend le modèle précédent en l'étendant à quatre niveaux. Elle intègre un niveau additionnel entre les niveaux Application et Réseau appelé niveau de Service de gestion. Ce niveau inclut le traitement de l'information par l'analyse, le contrôle de sécurité, la modélisation des processus et la gestion des périphériques. La proposition [16] considère l'architecture de base à trois niveaux mais en subdivisant le niveau Application en deux sous-niveaux. Le premier est le sous-niveaux Application qui implémente l'application métier en se basant sur les informations fournies par les niveaux sous-jacents. Le

deuxième est le sous-niveau Service qui intègre et stocke les informations depuis le réseau et réalise la gestion de ces informations, par exemple, l'analyse de données et la prise de décision.

Dans cette thèse, nous adoptons une vision architecturale considérant quatre niveaux (Figure 1.3) pour la séparation des différents rôles. Cette vision est proposée par le standard SmartM2M [17]. Elle considère les niveaux Application IoT, Middleware, Réseau et Équipements. Le rôle principal du niveau Middleware est d'assurer l'interopérabilité des applications et des objets en masquant les détails des technologies sous-jacentes aux applications [18]. Dans l'IoT, le Middleware abstrait l'hétérogénéité des réseaux et des équipements en fournissant une représentation homogène facilitant leur manipulation par les applications IoT. Ce rôle peut étendu via la capacité du Middleware à fournir d'autres services tels que la gestion des informations (par exemple, l'agrégation, le traitement et l'analyse des données), la gestion des services (par exemple, la découverte des objets et la configuration des services), ou encore la gestion des utilisateurs et des équipements.

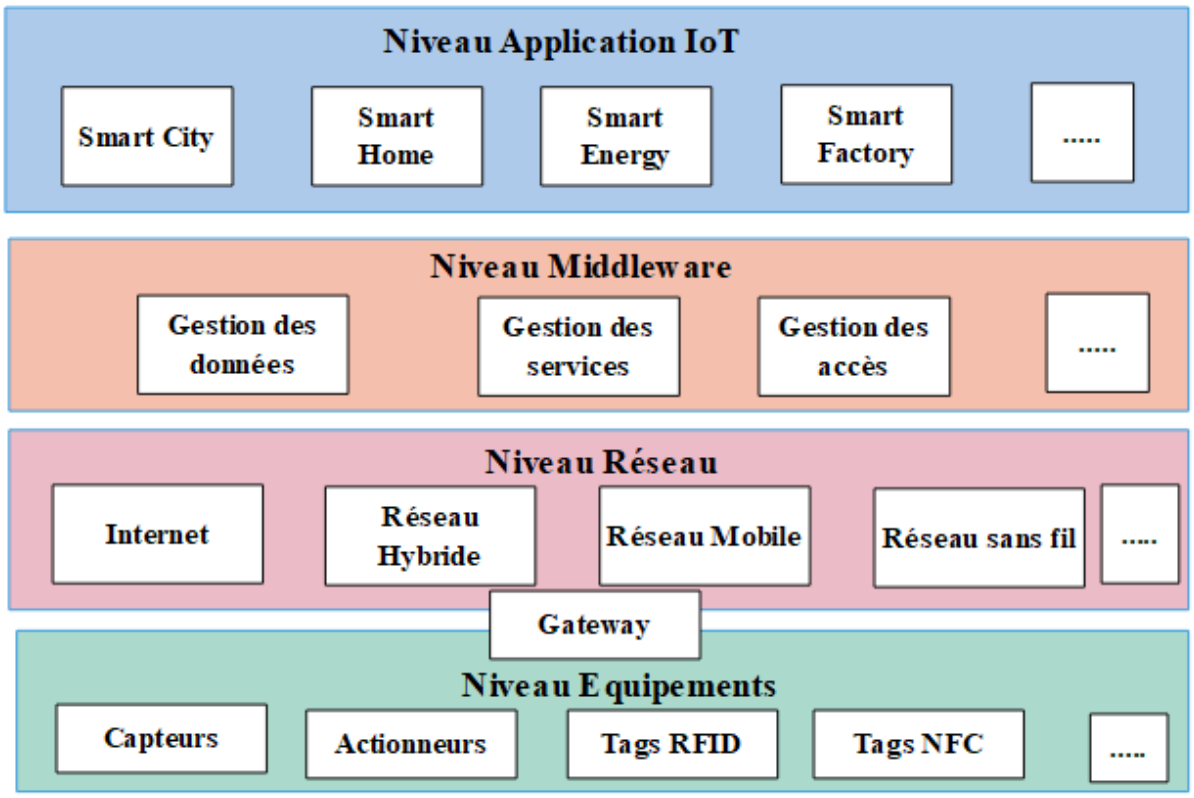

Figure 0.3: Modèle architectural pour IoT

Sur les différents niveaux de l'architecture de l'IoT, plusieurs challenges traditionnels doivent être reconsidérés. En effet, la spécificité de ce paradigme, liée notamment à la multitude des technologies utilisées au niveau Réseau et Équipements, rend inadaptées ou sous efficaces les

solutions proposées par ailleurs pour ces différents challenges. Nous citons trois challenges parmi les plus importants :

- **Fragmentation verticale et interopérabilité** : de nombreux industriels ont conçu et développé des solutions propriétaires dans le but d'amener de nouvelles applications IoT sur le marché. Cependant, cet état de fait a rapidement créé un problème de compatibilité entre les solutions des différents constructeurs, que ce soit au niveau architectural pour le lien entre les applications IoT et les équipements, ou au niveau interactionnel, où chaque application possède une API spécifique à la solution, créant ainsi un problème d'interopérabilité [19] entre les équipements et applications de différentes solutions ;
- **Sécurité et confidentialité :** l'IoT est caractérisé par l'utilisation d'équipements dits intelligents. Ces équipements génèrent des données à destination des applications ou reçoivent des commandes des applications pour contrôler leur environnement. En fonction du domaine applicatif, ces équipements doivent être de taille réduite et mobile, ceci à bas coût. Ces exigences peuvent rendre ces équipements ainsi que les réseaux vulnérables vis-à-vis des attaques externes. Ces attaques ciblent, par exemple, l'accès aux données, leur intégrité, le vol d'identité, ou le cambriolage [20], surtout lorsque les données communiquées sont susceptibles de contenir d'informations personnelles sur les utilisateurs et leurs habitudes [21]. De ce fait, de nouvelles techniques plus légères (pouvant être supporté par ces équipements) mais efficaces doivent être établies en adéquation avec le contexte de l'IoT ;
- **Scalabilité et Qualité de Service** : le terme de scalabilité désigne ici la capacité du système à s'adapter à la charge qu'il subit en maintenant les mêmes fonctionnalités et les mêmes performances indépendamment de cette charge. Dans l'IoT, le système peut avoir des problèmes de scalabilité vu la multitude des équipements et des applications qui échangent du trafic en passant par le cœur du système. La QoS requise par une application se traduit par plusieurs métriques telles que le temps de réponse, le taux de pertes ou encore la disponibilité qu'il s'agit de maintenir à un certain niveau pour la bonne exécution de l'application. Dans un contexte IoT et en fonction du domaine ciblé, les applications métiers peuvent présenter des besoins en QoS plus ou moins critiques qui doivent être pris en considération par les niveaux sous-jacents. Par exemple, dans le domaine du e-Health, la supervision et l'intervention en urgence dans les plus brefs

délais peut être primordiale face à la survie du patient. Nous parlons ainsi d'applications critiques.

Dans la section qui suit, nous présentons une caractérisation des applications métiers dans l'IoT. Elle concerne le type de données générées, le type d'interactions entre ces applications et les entités sous-jacentes du système, ainsi que les métriques de QoS liées ces applications.

1.3 Application métier dans l'IoT

Dans les différents domaines, plusieurs familles d'applications IoT peuvent être considérées en fonction de leurs profils (types de données, types d'interactions et besoins en QoS) et de leurs rôles (supervision, intervention, etc.). Nous retrouvons par exemple les applications dédiées à la télésurveillance en temps réel, la génération d'alarme et l'intervention en cas de problème.

De ces différents types d'applications IoT, plusieurs spécificités peuvent être extraites telles que le type de données générées, le type d'interactions entre les applications et les entités Middleware, ou encore les besoins en QoS qui reflètent la sensibilité de l'application vis-à-vis de métriques telles que le délai, les pertes, la disponibilité, etc. Dans cette section, nous présentons tout d'abord les différents types de données qui peuvent être générés dans le contexte de l'IoT. Nous présentons ensuite les types d'interactions entre les applications et les entités Middleware.

1.3.1 Types de données générées par les entités IoT

Dans un contexte IoT, les données sont échangées entre les différentes entités afin de mettre en place un service métier. Ces données peuvent être de différents types de par la diversité des équipements connectés (capteurs, actionneurs, caméras, microphones, cardiographe, etc.). Nous divisons les données échangées en deux types. Le premier concerne les données brutes issues d'équipements basiques (des relevés de température par exemple). Le deuxième type fait référence aux données multimédia (audio, vidéo) qui peuvent être issues d'équipements plus sophistiqués.

1.3.1.1 Les données brutes

Les données brutes sont transmises sous forme de texte simple transporté via un protocole de transfert. Dans un contexte IoT, ces données peuvent être issues d'équipements tels que des capteurs pour reporter par exemple la température, le degré d'humidité, la

localisation, le niveau d'un gaz, ou des signaux bio pour le diagnostic médical (fréquence cardiaque, respiration, etc.). Ces données sont véhiculées via des protocoles de transfert tels que http [22] basé sur le protocole de transport TCP, CoAP [23] qui s'appuie sur UDP, ou encore MQTT [24] qui repose sur TCP. La description de ces données se fait, par exemple, via des langages de balisage tels que XML [25] ou JSON [26]. Ces données peuvent être simples telles qu'une mesure d'un capteur, ou composées de plusieurs valeurs simples afin de constituer une information utile telle que la pression artérielle. Les données brutes peuvent aussi être issues des applications IoT. Elles sont dédiées aux commandes et envoyées sous forme de requêtes par les applications et peuvent servir, par exemple, à envoyer une requête vers un équipement tel qu'un capteur pour récupérer son état, ou pour contrôler à distance un actionneur ou un moteur.

1.3.1.2 Les données multimédia

Les données multimédia sont plus riches. Elles peuvent, par exemple, être constituées d'image, d'audio, et/ou de vidéo. Les données de type Image [17] peuvent, par exemple, être utilisées dans le domaine de la télésurveillance de patients pour le visionnage et l'analyse par le médecin d'une blessure, d'un symptôme, ou pour la prise d'une radio à distance.

Le trafic Audio [27] permet d'échanger du son (de type voix ou autre) entre les entités impliquées. Ce type de données est très utile dans le cas d'intervention en urgence, par exemple, lors d'un diagnostic de l'état d'un patient en situation critique, ou lors d'un accident routier [28]. Un flux audio de type voix est véhiculé via des réseaux privés ou d'opérateurs (LTE, 3G, etc.) passant par l'Internet, par le biais de techniques de VoIP [29] ou AoIP [30]. Ces techniques peuvent être basées sur des protocoles propriétaires ou standardisés. Des exemples de protocoles VoIP sont en majorité représentés par la figure suivante et peuvent être H.323, MGCP, SIP, H.248 (Megaco), RTP, RTCP, SRTP, SDP, IAX, etc.

Concernant le trafic vidéo dans l'IoT, il peut essentiellement être généré à des objectifs de surveillance à distance [31]. Il peut s'agir par exemple de télésurveillance du domicile ou bien la téléconsultation d'un patient auprès d'un médecin. Le flux peut être fait transmis en unicast (de 1 en 1), en multicast (de 1 vers N ou de N vers M) ou en broadcast (de 1 vers tous). Il peut être basé sur les protocoles RTP, RTCP (standards de l'IETF), MMS (propriété de Microsoft), Adobe RTMP (propriété d'Adobe Systems) ou de nouveaux formats non propriétaires qui commencent à émerger tels que MPEG-DASH pour le streaming à débit adaptatif sur http [32].

1.3.2 Types d'interactions entre les entités IoT

La génération des données précédentes se fait suite à l'interaction entre les entités du système IoT. Plusieurs types d'interactions [17] peuvent avoir lieu entre ces entités en fonction de la source et de la destination de la requête, mais aussi des informations que contiennent les données échangées.

Un premier type d'interaction concernent les données des équipements. Ces requêtes peuvent émaner des équipements (capteurs ou actionneurs) pour la sauvegarde de leurs données l'IoT (liaison montante). Ces requêtes sont émises de façon périodique ou événementielle suite à un changement d'état (Figure 1.4 - p1). Elles peuvent aussi provenir de l'application en vue de la récupération d'une valeur de capteur ou de l'état d'un actionneur. Deux cas de figure se posent alors : (1) la requête est adressée au système IoT pour la récupération d'une donnée stockée localement dans la base de données du Middleware (Figure 1.4 - p2), ou (2) la requête passe par le système IoT qui la redirige vers l'équipement concerné (Figure 1.4 – p3).

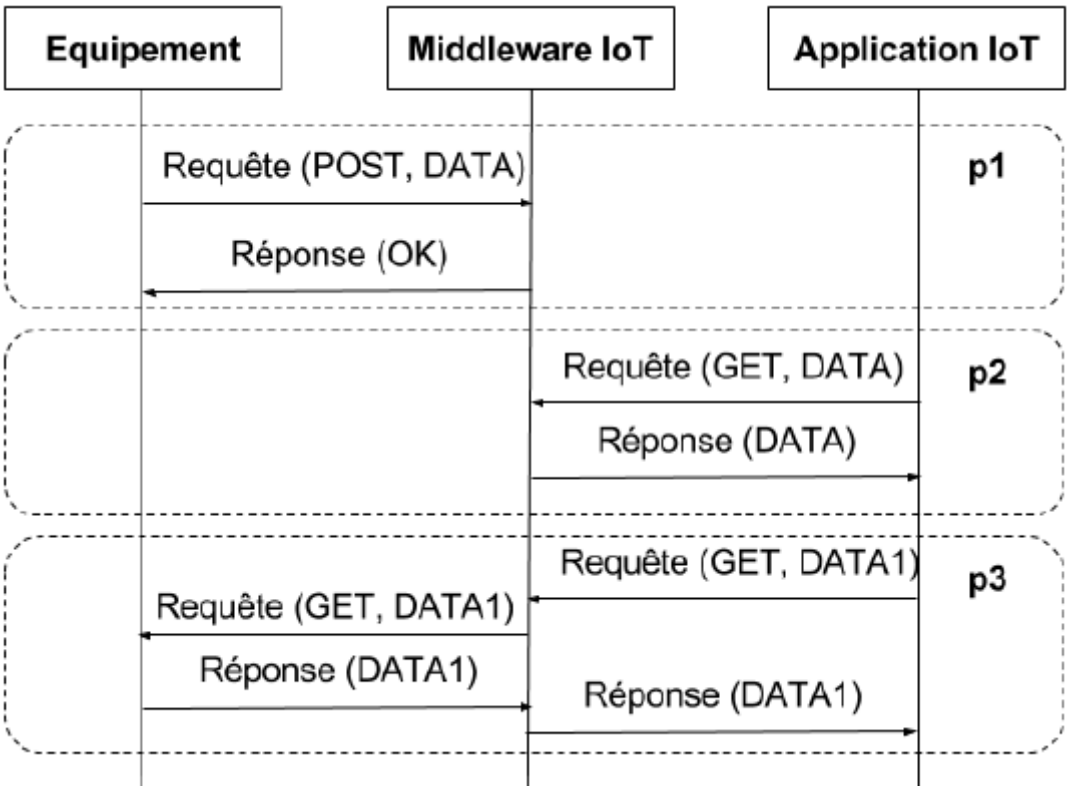

Figure 0.4: Interactions de types requetes / réponse pour le cas de données d'équipements

Un second type d'interaction se présente quand la requête provient d'une application pour le contrôle d'un équipement dans le cadre d'un scénario métier (Figure 1.5). La réponse confirmant la prise en compte de cette commande par l'équipement peut, par exemple, être son

état courant (ON / OFF) s'il contrôle une lampe, un ventilateur, etc., ou une autre forme de flux de données si la requête est dédiée au déclenchement d'une caméra ou d'un micro.

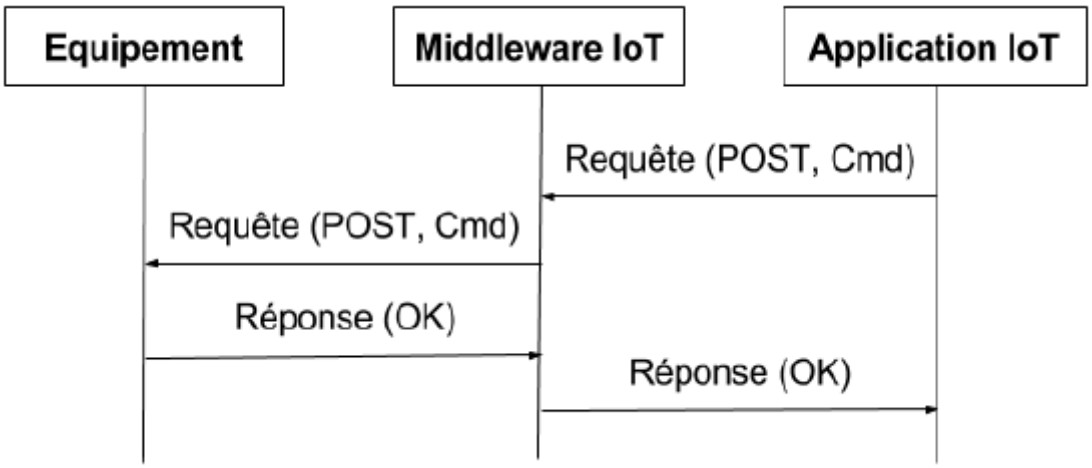

Figure 0.5: Interaction pour le contrôle d'un actionneur

Un troisième type d'interaction concerne la signalisation (Figure 1.6). Une requête de signalisation provient soit de l'application soit du Middleware pour la réalisation d'une authentification, d'un enregistrement ou d'une mise à jour du micro-logiciel (firmware) d'un équipement.

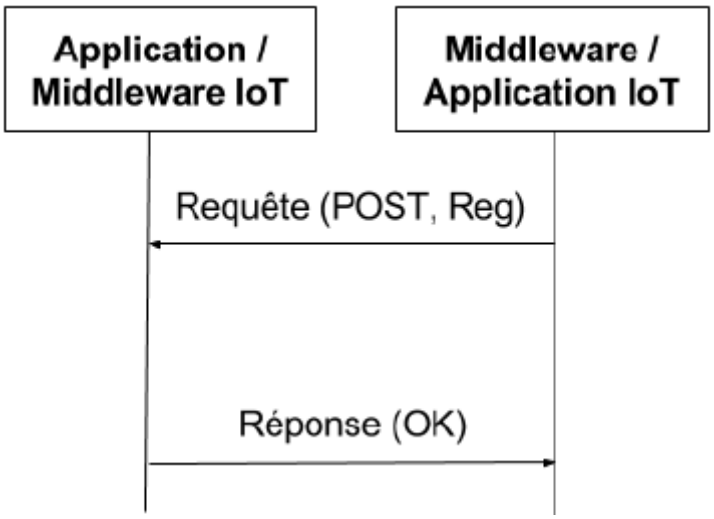

Figure 0.6: Interaction de signalisation entre l'entité Middleware et application IoT

Un autre type d'interaction entre l'application et le Middleware concerne la procédure de souscription / notification (Figure 1.7). Dans ce cas de figure, la souscription de l'application se fait sur toute nouvelle donnée issue d'un équipement. A la réception de cette donnée par le Middleware, ce dernier envoie une notification à l'application comportant la donnée qui vient

d'être créée. Cette donnée peut, par exemple, être une donnée d'un capteur ou un nouvel état d'un actionneur.

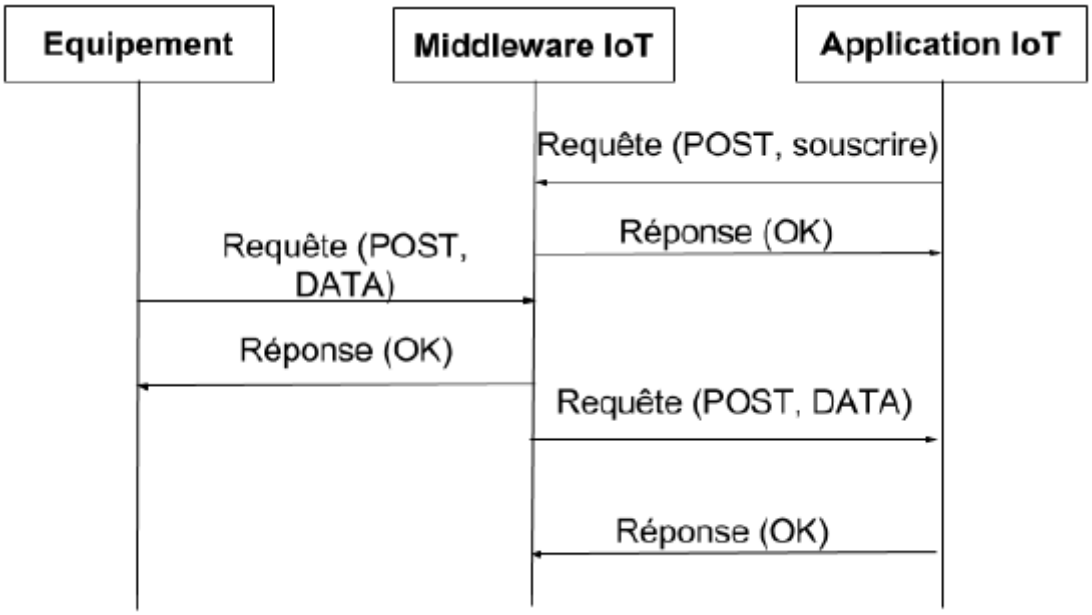

Figure 0.7: Interaction de type souscription / notification entre les entités IoT

D'autres types d'interactions plus poussées existent. Nous retrouvons par exemple le longpolling [33]. Dans ce cas, l'application envoie une requête à laquelle le serveur répond par une notification quand la donnée devient disponible. Après la réception de la donnée, l'application doit envoyer une nouvelle requête de long-polling vers le serveur et attendre la réponse de ce dernier. Du fait que la requête de l'application ne peut pas rester indéfiniment pendante, le serveur envoie une réponse vide avant l'expiration du temps de vie de la requête.

Cette réponse informe l'application du fait qu'il doit envoyer une nouvelle requête long-polling si elle veut continuer à recevoir les notifications. Il existe aussi le type time-based pull [34] où l'application envoie une requête de type pull pour une nouvelle donnée. Quand la donnée attendue devient disponible (par exemple, la mesure d'un capteur), le serveur envoie cette donnée à l'intérieur d'une réponse de notification. Sinon, il envoie une réponse vide. La fréquence des requêtes périodiques de l'application dépend de la définition de la fenêtre temporelle du pull (pull time window) qui doit être ajustée à la génération des données. Quand la valeur est trop petite, les requêtes de l'application peuvent arriver au serveur plus rapidement alors que les données ne sont pas encore prêtes (les ressources sont inutilement dépensées). Quand la fenêtre est trop large, l'application ne va pas acquérir toutes les données intéressantes.

1.3.3 Besoins en QoS des applications IoT

En fonction du contexte et du rôle d'une application IoT, celle-ci peut présenter un caractère critique et avoir des besoins en QoS. Ces besoins doivent être considérés et gérés par les niveaux sous-jacents du système IoT.

Parmi les besoins les plus importants, nous retrouvons [35] :

- **la disponibilité** : définie comme le taux de disponibilité opérationnelle d'un système ou d'un composant lorsqu'il est requis pour utilisation. Pour les applications IoT critiques (par exemple, la surveillance des signes vitaux des soins de santé), la disponibilité est une exigence importante que l'on souhaite être très proche de 100% ;
- **La fiabilité** : se réfère à la capacité d'un système ou d'un composant à exécuter ses fonctions dans des conditions données et pour une période de temps spécifiée. Dans l'IoT, les composants doivent être les plus fiables possibles pour la surveillance continue et la gestion des requêtes ;
- **La priorité :** peut être considérée comme l'importance attribuée à une requête ou sa réponse par rapport à une autre. Le trafic de l'entité IoT ayant une priorité plus élevée doit être traité en priorité par le système, suivant une politique de gestion appropriée ;
- **Le délai** : nous distinguons deux types de délai. Le premier est le délai d'aller-retour (RTT - Round Trip Time) qui correspond au temps nécessaire pour répondre à une demande. Il est considéré comme la somme du temps d'envoi de la demande et de la réponse, ainsi que du temps de traitement de différents composants parcourus dans le système IoT. Le deuxième type est le délai de bout en bout correspondant, par exemple, au transfert de données entre l'équipement et l'application IoT. Par exemple, [36] a fixé un délai maximal de 300 ms pour le transfert d'une électrocardiographie en temps réel ;
- **La périodicité** : définie en rapport avec l'intervalle temporel d'exécution des actions. Au-delà de cet intervalle, la donnée devient obsolète. La périodicité est une caractéristique spécifique des applications IoT de surveillance. Les données collectées peuvent avoir différentes périodes selon l'activité métier ;
- **L'intégrité des données** : garantit la non altération des données, que ce soit par accident ou par des parties non autorisées. Les données collectées provenant des équipements ne doivent pas être modifiées pour assurer la réalisation de l'action correcte par l'application IoT ;
- **La confidentialité :** liée à l'assurance que les données ou le système IoT ne peuvent être accessibles que par ceux qui en ont l'autorisation. Cette propriété est essentielle

dans ces systèmes qui doivent garantir que les informations des équipements ou machines ne sont accessibles qu'aux parties autorisées.

Les applications IoT Micro ainsi être classifiées selon plusieurs critères. Le Tableau 1.1 illustre une classification de quelques applications IoT relatives aux domaines du e-Health et Smart Grid. Cette classification prend en considération le débit exigé, la sensibilité au temps, ainsi que d'autres besoins en QoS.

Tableau 0.1: Classification des Applications IoT

Domaine	**E-Health**			**Smart Grid**	
Application	Télédiagnostic	Téléconsultation	Télémonitoring	Smart metering	SCADA
Type de données	Brut et multimédia (image)	Multimédia (audio et vidéo)	Brut et multimédia (image et vidéo)	Brut	brut
Temps réel	~	Oui	~	Non	Oui
Débit	Elevé	Elevé		Faible	Moyen
Périodicité	Non	Non	Oui	Oui	
Temps de réponse	Oui	Oui	~	Non	Oui
Fiabilité	Oui	Oui	Oui	Oui	Oui
Disponibilté	Oui	Oui	Oui	Non	Oui
Priorité	~	Oui	~	Non	Oui
Deadline	~	Oui	~	Non	~

Dans ce tableau, nous considérons deux domaines (e-Health et Smart Grid). Nous illustrons trois applications du domaine du e-Health [37] [38]. La première est une application de télédiagnostic véhiculant des données brutes (texte) ou multimédia (images) issues de capteurs et d'équipements (ECG). Elle peut être temps réel en transférant des images médicales à une localisation distance dans des situations d'urgence, ou non temps réel dans le cas, par exemple, d'une simple analyse de données par des spécialistes. La deuxième application est une application de téléconsultation se fait en temps réel sur la base de conversation entre docteur / patient ou entre docteur / docteur. Enfin, la troisième application est une application de télé-monitoring peut être temps réel en cas de transmission des signes vitaux du patient et vidéo streaming en cas d'urgence, ou non temps réel quand la transmission des signes vitaux se fait du patient vers le centre de supervision pour une analyse.

Pour le domaine du Smart Grid, nous illustrons deux applications. La première application, de smart metering, véhicule des données brutes qui n'ont pas d'exigences de temps réel, de débit, de temps de réponse, de disponibilité de priorité ou de deadline. La deuxième application, SCADA (Supervisory Control and Data Acquisition), a plus d'exigence notamment en temps réel puisqu'elle doit assurer le contrôle à distance des installations techniques dans des environnements contraints tels que l'industrie. La fiabilité est une exigence pour toutes ces applications. Par rapport au débit des données, trois valeurs sont considérées : faible, moyen ou élevé. Il est considéré comme étant faible lorsqu'il est inférieur à 12 kbps, moyen s'il est entre 12 kbps et 24 kbps, et élevé quand il est supérieur à 24 kbps [37].

Les besoins applicatifs en terme de QoS peuvent être différents en fonction du contexte et du rôle joué par l'application métier (supervision, intervention, etc.). Afin d'assurer la prise en compte de ces besoins, ils doivent être considérés par les différents niveaux sous-jacents du système IoT. Dans cette thèse, nous traitons la gestion de la QoS au niveau Middleware, qui constitue le premier niveau pour la prise en compte de ces besoins. Nous abordons dans ce qui suit, les différentes solutions Middleware, ainsi que les propositions faites pour la gestion de la QoS à ces niveaux.

1.4 Solutions au niveau middleware et gestion de la QoS

Dans le contexte de l'IoT, le Middleware permet d'abstraire les applications de la complexité des niveaux sous-jacents. Cette complexité est due à la diversité et l'hétérogénéité des technologies réseaux et équipements. Au sein de l'IoT, les modes d'interaction peuvent suivre plusieurs styles architecturaux selon l'implémentation du Middleware. Dans cette section, nous détaillons tout d'abord ces différents styles architecturaux. Nous exposons ensuite les solutions les plus abouties issues d'organismes de standardisation, d'alliances ou de contributions isolées. Enfin, nous nous focalisons sur les propositions de gestion de la QoS au niveau Middleware dans le contexte de l'IoT et au-delà.

1.4.1 Styles architecturaux pour le middleware

Un système, en particulier logiciel, est défini par son architecture. Celle-ci comporte des éléments structurels (ici des composants logiciels), leur comportement (i.e. l'algorithmique interne des composants), leurs interfaces, leur composition via des liens d'interconnexion, les objets échangés via ces liens, ainsi que le style architectural qui va régir cette organisation [39]. Un style architectural définit une famille de systèmes en termes d'organisation structurelle et

de vocabulaire de composants et de connecteurs [40]. Dans cette section, nous définissons les principaux styles architecturaux envisagés pour l'implémentation du Middleware. Chacun d'eux est analysé sous l'angle de ses performances et de ses capacités à répondre aux besoins en gestion de la QoS.

Dans cette section, nous définissons les différentes approches des styles architecturaux permettant d'implémenter le Middleware.

1.4.1.1 Remote Procedure Call

Dans les systèmes distribués, le modèle RPC (Remote Procedure Call) permet la communication entre processus distants [41]. Il permet à une application d'exécuter une procédure offerte par un serveur distant [42]. Le serveur est vu comme un ensemble de procédures exécutables via des appels par un client distant afin de réaliser une tâche spécifique. Parmi les protocoles implémentant le modèle RPC, le plus utilisé est le système NFS (Network File System) développé par SUN Microsystems qui fait partie de la couche application [43].

Etant donné que la logique métier est centralisée au niveau d'un serveur central, le modèle RPC présente des limites. En terme de QoS, elle résulte de celle offerte par les protocoles des couches sous-jacentes à savoir les couches Transport et Réseau. Le modèle RPC ajoute aussi une complexité supplémentaire en utilisant le protocole HTTP comme simple protocole de transfert sans exploiter les méthodes qu'il offre, alourdissant ainsi le corps de la requête en y incluant les appels de procédure [44] et en impactant ainsi les performances du système.

1.4.1.2 Approche orientée objets

Cette approche se base sur l'utilisation d'un ORB (Object Request Broker) afin de permettre l'interaction distante avec des objets déployés dans un même espace mémoire local [45]. Ces objets communiquent entres eux par l'envoi et la réception de requêtes et de réponses, de manière transparente et portable. Cette approche représente ainsi une extension de l'approche RPC en offrant davantage d'interopérabilité. Plusieurs implémentations de cette approche existent. Nous retrouvons par exemple parmi les plus connues : CORBA [46], qui est un standard défini par le groupe OMG pour les ORB, .NET Remoting pour les plateformes Microsoft, et RMI qui est une API Java développée par SUN Microsystems et qui permet l'invocation distante de méthodes Java [47] [48].

Malgré l'interopérabilité et l'intégrabilité que permet d'offrir l'approche orientée objets, les besoins en QoS ne sont pas adressés par les solutions basiques. Les solutions plus avancées (telle que CORBA) supportent une certaine QoS en terme de tolérance aux fautes [49].

1.4.1.3 Approche orientée messages

Les Middlewares orientés messages (MOM - Message-Oriented Middleware) supportent l'envoi et la réception de messages entre systèmes distribués. Cette approche est basée sur le concept de message et de canal de communication qui peut être de 'un vers un', de 'un vers plusieurs', de 'plusieurs vers un', ou de 'plusieurs vers plusieurs', ainsi qu'un agent appelé Message broker [50]. Parmi les implémentations les plus utilisées du Message broker, nous retrouvons : Apache ActiveMQ [51], Fuse [52], RabbitMQ [53], ainsi que le protocole AMQP (Advanced Message Queuing Protocol) pour traiter les problématiques d'interopérabilité [54].

L'approche MOM réduit la complexité des systèmes distribués hétérogènes. En ce qui concerne la QoS, plusieurs implémentations telles que DDS (Data Distribution Service for Real-Time Systems) [55] ou RabbitMQ permettent de la gérer via, par exemple, des mécanismes de persistance de données pour assurer la disponibilité des messages et la fiabilité de leur transfert.

1.4.1.4 Approche orientée services

L'approche orientée services permet d'élaborer des architectures dites SOA (Service Oriented Architecture) [56] pour les systèmes distribués. Cette approche se base sur l'échange de données entre les entités de l'architecture en tant que services web. Les solutions basées SOA garantissent l'interopérabilité en se basant sur des protocoles tels que HTTP, JMS, SMTP ou FTP pour la communication entre des plateformes hétérogènes. Les bus de services ou ESB (Enterprise Service Bus) [57] constituent l'une des implémentations les plus connues de SOA. L'ESB joue le rôle de Middleware entre producteurs et consommateurs de services. Il permet l'échange asynchrone via des interfaces telles que SOAP, JMS ou JBI.

Les solutions orientées SOA sont toutefois limitées. Les protocoles de communication (par exemple HTTP) ne sont exploités que pour véhiculer les messages. Ces messages contiennent la représentation ainsi que la méthode invoquée, ce qui nécessite à chaque fois la désencapsulation du corps du message pour déduire la méthode. Il ne tire donc pas partie de la méthode utilisée par le protocole de communication et peut engendrer des problèmes de QoS.

1.4.1.5 Approche orientée ressources

L'approche orientée ressources est basée sur les principes du style architectural REST (Representational State Transfer) [58]. Ce style architectural considère que toute entité physique ou logique est une ressource ayant une représentation accessible à distance. Chaque ressource est adressable de manière unique via un URI (Uniform Resource Identifier). Les systèmes RESTful se basent essentiellement sur des protocoles de transfert tels que HTTP ou CoAP tout en exploitant leurs méthodes pour alléger le corps de la requête.

REST offre donc une grande interopérabilité en exploitant les méthodes du protocole de transfert (par exemple GET, POST, PUT, DELETE pour le protocole HTTP). Il est aussi sans état, ce qui le rend scalable puisque le serveur ne gère plus l'état des sessions des utilisateurs.

Pour pallier au problème de fragmentation verticale des solutions IoT, plusieurs solutions Middlewares ont été proposées que nous présentons à présent. Dans ce qui suit, le focus est plus spécifiquement porté sur les propositions issues des organismes de standardisation et des alliances.

1.4.2 Propositions de niveau Middleware dans l'IoT

Plusieurs solutions ont été proposées au niveau Middleware afin de palier au problème de fragmentation verticale de l'IoT. Nous regroupons ces propositions en deux familles : celles issues des organismes de standardisation qui proposent les cadres architecturaux, les composants fonctionnels et les interactions entre eux ; et celles basées sur des alliances qui proposent directement des plateformes Middleware open source.

Dans cette section, nous commençons tout d'abord par la première famille en présentant les standards SmartM2M et oneM2M. Ensuite, nous passons aux alliances avec la présentation des projets open source IoTivity et FIWARE.

1.4.2.1 Standard SmartM2M

L'ETSI (European Telecommunication Standard Institute) a proposé une première version du standard SmartM2M en 2011 [17]. Il a établi un standard complet comprenant l'architecture fonctionnelle, les interfaces de communication, la représentation et la structuration des données, ainsi que toutes les procédures nécessaires entre entités du système d'une part, et entre ces mêmes entités et les entités externes au système. La couche d'abstraction, appelée SCL (Service Capability Layer), propose un ensemble de services de

niveau Middleware, tels que la communication générique, l'accessibilité, l'adressage et le stockage, la sélection du canal de communication, la gestion des entités (équipements, données, droits d'accès, etc.), la sécurité, etc. Cette couche de service peut être déployée sur un serveur de concentration (NSCL - Network SCL) avec lequel interagissent les applications métier, ou sur des gateways (GSCL - Gateway SCL) reliées au serveur et permettant l'interconnexion des équipements non capables de communiquer avec l'extérieur.

L'interaction entre les entités de l'architecture se fait via des interfaces de communication basées sur le style architectural REST. Il donne une représentation sous forme de ressources des différents services. Ces ressources sont adressables via une URI et gérables via les méthodes CRUD (Create, Read, Update et Delete).

Le standard donne une structuration de ces ressources sous forme d'une représentation arborescente appelée arbre de ressources. Cet arbre vise à offrir des fonctions de médiation de données en tant que moyen de simplifier l'adressage et la maintenance des ressources, décrire comment les différents types de ressources sont liés entre eux et améliorer la performance globale du système grâce à l'utilisation de données minimalement structurées. Ce style permet ainsi aux applications IoT d'interagir indépendamment du type de technologies réseau et d'équipements sous-jacents. Le standard propose également l'intégration de protocoles tels que HTTP et CoAP pour la communication distante entre les différentes entités du système.

Vis-à-vis des besoins de QoS, le standard SmartM2M n'intègre aucun service au niveau Middleware permettant de prendre en considération ces besoins. Il considère que la QoS est celle résultante de la couche sous-jacente (réseau).

1.4.2.2 Initiative oneM2M

L′initiative oneM2M [59] s′inscrit dans une vision globale d′organismes de standardisation pour offrir un unique standard. Elle propose également une couche de services RESTful horizontale. Cette couche permet l′enregistrement mutuel entre les entités, la découverte des services et des ressources, la gestion des équipements, la sécurité, etc. Nous la considérons généralement comme le successeur de SmartM2M.

L'architecture oneM2M est composée de trois couches. La couche Application est constituée des entités applicatives (AE - Application Entity) qui représentent les applications interagissant avec le serveur, les gateways ou les équipements. La couche Service, appelée CSE

(Common Service Entity), représente la couche d'abstraction Middleware. La couche Réseau englobe tous les réseaux de communication et équipements sous-jacents. Le standard repose aussi sur la notion de ressources et propose le même arbre de ressources que SmartM2M, avec intégration de protocoles de communication tels que HTTP, CoAP, ou MQTT.

1.4.2.3 Projet IoTivity

IoTivity est un projet open source financé par l'OCF (Open Connectivity Foundation) [60] [61]. Il propose un framework permettant la connectivité entre les équipements D2D (Device-to-Device) afin de répondre aux besoins de l'IoT en terme d'interopérabilité. Le framework offre des services aux applications IoT tels que la gestion et la découverte d'équipements, la transmission et la gestion de données. Plusieurs protocoles sont supportés, nous retrouvons par exemple CoAP, basé sur le style architectural REST, mais aussi Zigbee, Z-wave, Bluetooth, etc. En 2016 ce projet a été rejoint par le projet [62] de l'alliance AllSeen. AllJoyn propose un framework basé sur l'approche producteur / consommateur pour l'échange de données.

1.4.2.4 Projet FIWARE

FIWARE [63] est un projet conduit par l'union européenne dans le cadre de l'IERC [64]. Il propose une plateforme open source de niveau Middleware qui offre un écosystème pour la création de nouvelles applications et services pour la mise en place de scénarios métier et de fonctions de l'IoT. La plateforme offre des fonctionnalités Cloud améliorées basées sur le framework OpenStack (telles que l'allocation de ressources, le provisionnement, le stockage, l'orchestration, etc.) ainsi qu'un ensemble d'outils et de bibliothèques connus sous le nom de Générique Enablers (GEs). Ces GEs offrent différentes capacités telles que la délivrance des applications, services et équipements, la gestion des données et du contexte, fourniture d'interfaces avec les réseaux et équipements, le stockage en cloud, la sécurité, etc.

Un nouveau service qu'offre les GEs de l'architecture FIWARE est celui de la gestion du contexte. Ce service propose un mécanisme pour générer, collecter, publier, demander ou analyser des informations (potentiellement massives) de contexte de manière efficace. Ces informations sont utilisées par les applications afin de réagir à leur contexte. Elles peuvent par exemple être des informations de localisation, de présence, liées au profil de l'utilisateur ou aux terminaux. Ce processus est complexe car ces informations peuvent provenir de différentes sources, systèmes, applications, capteurs, etc.

1.4.3 Gestion de la QoS au niveau Middleware

Dans le contexte de l'IoT, en plus des besoins fonctionnels que doit remplir le Middleware en abstrayant la complexité des couches sous-jacentes, d'autres besoins non fonctionnels peuvent être exprimés par les applications IoT et doivent donc être pris en considération, notamment le besoin en QoS. Dans cette section, nous présentons dans un premier temps les travaux proposant des solutions au niveau Middleware pour la gestion de la QoS dans le contexte de l'IoT. Ensuite, nous présentons quelques-unes des principales solutions au niveau Middleware dédiées à d'autres contextes, toujours dans une perspective de gestion de la QoS.

Solutions Middleware orientées IoT pour la gestion de la QoS

Dans le cadre des efforts de standardisation, plusieurs couches de services de niveau Middleware ont été proposées dans le contexte de l'IoT. Ces couches intègrent plusieurs services fonctionnels et non fonctionnels pour la gestion des équipements IoT. Cependant, ces standards ne proposent aucune solution de gestion de la QoS au niveau Middleware [65]. Ils considèrent que la QoS résulte de celle fournie par les réseaux sous-jacents.

Plusieurs solutions spécifiques (i.e. hors standard) ont cependant été proposées. Dans [66] et [67], les auteurs proposent d'améliorer le Middleware WuKong afin de garentir la gestion de la QoS. Ils introduient la notion de score de qualité qui prend de multiples métriques de QoS (temps de réponse, fiabilité, etc.) en considération. Il sélectionne les équipements physiques adéquats et décide du déploiement des équipements le plus optimal, c'est-à-dire engendrant le plus grand niveau de score. La limite de l'approche est liée au fait que les besoins en QoS des différentes applications ne peuvent pas être prises en compte dynamiquement.

Dans le projet MiLAN [68], Heinzelman propose un Middleware qui gère le réseau et les noeuds. En fonction de la description de l'application et de ses besoins en QoS, le Middleware configure le réseau et les noeuds pour satisfaire ces besoins. Ceci étant, MiLAN a besoin d'un diagramme d'états spécifique à chaque scénario applicatif et au contexte de réseau de capteurs sans fil, ce qui est relativement plus compliqué vu les changements dynamiques dans le contexte de l'IoT.

D'autres solutions telles que [69] reposent sur l'intégration du protocole MQTT pour la gestion de la QoS [70]. Ce protocole inclut une forme basique de gestion de la QoS. Il propose trois niveaux de garantie de l'arrivée du message. Le niveau QoS 0 offre un mode best effort

sans aucun acquittement. Le niveau QoS 1 garantit la délivrance du message au récepteur au moins une fois. Le niveau QoS 2 garantit la délivrance unique du message au récepteur. Enfin, les dernières spécifications du standard oneM2M proposent l'intégration du protocole MQTT [60]. Face au problème de la QoS, ce protocole offre une certaine garantie pour la délivrance du message. Cependant, dans un contexte IoT, les applications métier peuvent avoir des besoins différents et plus complexes que la simple garantie de délivrance du message.

1.5 WoT

Le Web des objets (WoT) peut être considéré comme une spécialisation de l'Internet des objets (IoT), où, d'une part, toute la complexité de la partie de la connectivité des objets intelligents est abstraite, et d'autre part, elle fournit une couche d'application standard basée sur les normes Web pour simplifier la création d'applications IoT. Dans l'IoT, le protocole de communication unique à application unique submerge, ce qui crée des silos d'utilisateurs et limite l'exploitation du plein potentiel de l'IdO. L'un des principaux intérêts de l'utilisation du WoT au lieu de l'IoT réside dans les différents avantages qu'il offre. Il convient de mentionner la simplicité de développement, le couplage libre, puisque HTTP est couplé de manière libre par la conception, et l'ouverture des normes. L'idée est que tous les objets intelligents peuvent communiquer à l'aide d'un langage Web (HTTP) en exposant une API REST. Cette API peut être présente dans l'objet intelligent lui-même ou dans un intermédiaire pouvant agir pour le compte de l'objet intelligent afin d'exposer l'API Web [71]. Ceci est devenu heureusement possible avec l'amélioration des systèmes embarqués.

Par conséquent, trois solutions principales peuvent être distinguées afin de permettre aux dispositifs sous contrainte et non soumis à contrainte d'accéder à Internet et de fournir des services:

Solution 1: en implémentant directement l'API WoT sur l'objet intelligent lui-même. Techniquement, en exécutant un petit serveur Web sur l'objet intelligent. Par conséquent, l'objet intelligent peut communiquer via HTTP et fournit un accès à ses ressources via une API REST. De nos jours, nous pouvons faire fonctionner de minuscules serveurs à l'intérieur des appareils contraints tels que lighttpd [1], Nginx [2] et MiniWeb [3]. Cependant, cette solution n'est pas destinée aux appareils qui ne sont pas alimentés par batterie, car ils utilisent un serveur et / ou utilisent WIFI ou Ethernet l'appareil nécessite de l'énergie supplémentaire.

Solution 2 : concerne principalement les appareils sous tension alimentés par batterie et qui utilisent des protocoles basse consommation pour communiquer. Au lieu d'avoir l'API WoT sur le périphérique, celle-ci sera hébergée sur un intermédiaire capable d'exposer les fonctionnalités du périphérique (comme un routeur WIFI). Ces intermédiaires sont appelés passerelles ou passerelles d'applications et ne sont généralement pas contraints. D'un côté, ils peuvent communiquer avec des objets intelligents à l'aide de protocoles non Web, tels que Bluetooth, Zigbee, etc., et de l'autre côté, ils exposent l'API WoT de l'objet intelligent via une API REST. En plus de la conversion d'un protocole à un autre, ces passerelles peuvent également ajouter une couche de sécurité, stocker temporairement des données, afficher la description sémantique des objets intelligents, etc.

Solution 3 : est en utilisant le nuage. Cette solution est une extension de la solution de passerelle où, au lieu de placer l'API WoT sur un simple intermédiaire, la passerelle est un serveur cloud distant. Cette solution est utilisée dans les cas où il faut gérer une grande quantité de périphériques et de données et où une base de données plus puissante et plus grande est nécessaire.

Différence entre WoT et IoT

Afin de justifier la nécessité du mot d'ordre et de clarifier les différences entre la notion de l'Internet des objets et la notion de WoT [71], une liste d'arguments est fournie ci-dessous :

- **Plus une application-un protocole :** dans l'IoT actuel, pour chaque objet intelligent, une application dédiée pouvant communiquer exclusivement avec cet objet intelligent (ou avec un petit ensemble d'objets intelligents) est nécessaire, et utilisant généralement un protocole de communication spécifique. L'inconvénient de cette approche est que, d'une part, l'utilisateur devra gérer une application différente pour chaque objet intelligent et, d'autre part, il crée des silos d'utilisateurs et d'objets intelligents. Cependant, avec l'idée de WoT, une seule application Web peut contrôler tous ces objets intelligents, car chaque objet intelligent expose une API WoT.
- **Plus facile à programmer :** dans l'IoT, de nombreuses solutions et protocoles sont proposés pour connecter les différents objets intelligents. Par conséquent, l'apprentissage de la spécification de chaque protocole est une tâche ardue, en particulier pour les amateurs qui souhaitent connecter différents objets (cas de la maison intelligente). Cependant, avec WoT, la création d'une application Web pour connecter différents périphériques devient plus facile et plus rapide, car la plupart des

développeurs ont déjà une connaissance des protocoles Web. De plus, si tous les appareils pouvaient offrir une API WoT, les développeurs pouvaient utiliser le même modèle de programmation pour interagir avec l'un d'entre eux, ce qui peut être très intéressant dans le cas de la maison intelligente, des bâtiments intelligents et des villes intelligentes.

- **Normes ouvertes et extensibles :** dans l'IoT, les protocoles et les normes de communication sont continuellement proposés par différentes entités. Cependant, tous les protocoles ne sont pas publiquement disponibles pour la communauté. De plus, à tout moment, des modifications importantes peuvent être introduites dans le protocole, ce qui nécessite au moins une mise à jour du microprogramme, voire une modification du matériel. D'autre part, les standards Web ouverts et libres ont considérablement évolué. Par conséquent, il n'y a pratiquement aucun risque qu'ils changent du jour au lendemain.
- **Déploiement, maintenance et intégration rapides et faciles** : dans l'IoT, si une seule couche d'application est requise, des efforts et des investissements importants sont nécessaires pour écrire des convertisseurs personnalisés pour chaque nouvel appareil ou application devant être intégré au système. De plus, l'ajout de la limitation précédente, selon laquelle le protocole est modifié ou mis à niveau, ajoute à la complexité. Dans le WoT, il n'est pas nécessaire de maintenir ou de mettre à niveau le Web.
- **Couplage lâche entre les éléments :** dans l'IoT, un couplage très étroit entre l'appareil et l'application correspondante dans le réseau est présent. De plus, les différentes interactions possibles de l'appareil sont planifiées statiquement à l'avance.

  Par conséquent, il ne laisse pas beaucoup de place pour des interactions ponctuelles imprévues et la réaffectation des services dans de nouveaux cas d'utilisation. Pour le WoT, HTTP est faiblement couplé par conception. L'un des objectifs du WoT est de pouvoir ajouter n'importe quel périphérique au système et de pouvoir communiquer avec lui sans aucune mise à niveau.
- **Mécanismes de sécurité et de confidentialité :** l'un des principaux problèmes, que ce soit dans l'IoT ou dans le WoT, afin de créer une application réelle qui peut être sûre pour les utilisateurs et qui protège leurs données de toute menace. Le problème de l'IoT est que les mécanismes de sécurité sont parfois construits à partir de rien spécialement pour un appareil ou une application spécifique et parfois même n'existent pas. De plus, même ces mécanismes ne sont pas toujours sérieusement testés et peuvent présenter des failles sérieuses (clés faibles, mots de passe faibles, algorithmes d'échange de clés

faibles, clés statiques ou clés échangées publiquement, etc.). Dans le cas du WoT, la sécurité de la couche d'application sur le Web a suffisamment mûri pour résister à plusieurs attaques et des progrès considérables ont été accomplis pour fournir des mécanismes de sécurité fiables.

1.6 CONCLUSION

Ce chapitre a présenté le cadre général de notre travail. La première partie a été consacrée à l'IoT et ses applications. Nous avons démontré l'apport de l'IoT dans des contextes traditionnels. Nous avons ensuite donné les différentes visions architecturales de l'IoT. Celle que nous retenons est constituée de quatre niveaux : Application Métier, Middleware, Réseau et Équipements. Nous avons ensuite posé la problématique de la QoS. Le besoin provient d'une analyse des caractéristiques des applications IoT au travers des types de données qu'elles manipulent (brutes et multimédia). Enfin, nous avons défini le WoT et sa différence avec l'IoT.

2 Outils et protocoles de mutualisation des ressources dans les Middlewares IoT

2.1 Introduction

Dans ce chapitre, nous faisons une étude sur les protocoles et outils permettant de concevoir une plateforme de mutualisation. Ainsi, cette plateforme doit prendre en considération les normes et exigences des bonnes pratiques en matière de middleware IoT.

Nous examinerons les outils et protocoles de communication qui pourront assurer les échanges entre les utilisateurs pendant une session de consultation à distance, Télé-TP, etc.

Aussi, nous aborderons les protocoles assurant l'acquisition des données générées par les capteurs à distance des dispositifs de l'IoT ainsi que les infrastructures et les technologies qui contribueront à construire notre stratégie de mutualisation.

Pour terminer, nous ferons le choix du type de serveur WebRTC a utilisé pour le middleware IoT qui convient à notre architecture de mutualisation. Ce choix sera fait sur la base des bonnes pratiques de qualités de services sécuritaires d'une plateforme IoT intégrant IMS.

2.2 Etude des protocoles et normes

2.2.1 Protocoles d'applications IoT

Les outils de développement et de prototypage actuels (Arduino, Raspberry Pi, etc.) permettent de proposer des solutions innovantes à faible coût ayant pour conséquence un foisonnement de projets dans le domaine de l'IoT. En effet, pour peu de moyens, il est possible aujourd'hui de mettre en place un espace intelligent afin de fournir des services à partir des données recueillies par les objets connectés.

Cependant, pour un déploiement massif de l'IoT, les chantiers sont encore nombreux : respect de la vie privée, sécurité, autonomie des équipements, analyse des masses de données générées (Big Data), sans oublier la standardisation qui demeure encore aujourd'hui le talon d'Achille de l'IoT.

Avec le développement de l'IoT au cours de ces dernières années, les protocoles applicatifs se sont multipliés au point que cela devient difficile pour un néophyte de s'y retrouver, et de savoir par où commencer. Ainsi, en 2014, les groupes de travail sur l'internet des objets

(ISO/IEC JTC 1/WG 10) ont établi un projet standard pour l'architecture de référentiel de l'IoT. Afin de faire progresser efficacement les travaux en cours : les lacunes de normalisation, les technologies de niveau réseau pour l'IoT, l'identification de l'IoT, les directives d'intégration du système et le modèle de référence conceptuel pour l'architecture de référence de l'IoT.

Le Tableau 2.1 ci-dessous présente un résumé des protocoles les plus importants définis par ces groupes.

Tableau 2.1: Activités de normalisation à l'appui de l'IoT

Application	Service discovery		Protocoles d'infrastructures								
			L4	L3	L3	L2	L1	L1	L1	L1	L1
Prototocole	mDNS	DNS-SD	RLP	6LoWPAN	IPv4/IPv6	IEEE.802.15.4, 802.11, 802.3	LTE-A	EPC Global	IEEE.802.15.4	Z-Wave	IMS
DDS	oui	non	oui	non	non	oui	oui	non	non	non	non
CoAP	oui	non	oui	non	non	oui	non	oui	non	non	non
AMQP	oui	non	oui	non	non	oui	non	oui	non	non	non
MQTT	oui	non	oui	non	non	oui	non	non	oui	non	non
MQTT-SN	non	oui	oui	non	non	oui	non	non	oui	non	non
XMPP	non	oui	oui	oui	oui	oui	non	non	non	oui	non
HTTP, REST	non	oui	oui	oui	oui	oui	non	non	non	oui	non
SIP	non	oui	oui	oui	oui	oui	non	non	non	non	oui

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 des routes et échanger les données selon l'ensemble des règles du protocole applicatif sélectionné.

Nous avons résumé sur le tableau 2.2 les protocoles applicatifs IoT en fonction de leurs principales caractéristiques. La plupart de ces protocoles utilisent TLS ou DTLS comme mécanismes de sécurité [72].

Tableau 2.2: Comparaison des principales caractéristiques des protocoles de la couche applicative

Protocole	**Req.-Req.**	**Pub.-Sub.**	**Standard**	**Transport**	**QoS**	**Sécurité**
REST HTTP	oui		IETF	TCP	-	TLS/SSL
MQTT		oui	OASIS	TCP	3 levels	TLS/SSL
CoAP	oui	oui	IETF	UDP	Limité	DTLS
AMQP	oui	oui	OASIS	TCP	3 levels	TLS/SSL
DDS		oui	OMG	TCP/UDP	Exetensive	TLS/DTLS/DDS sec.
XMPP	oui	oui	IETF	TCP	-	TLS/SSL
HTTP/2.0	oui	oui	IETF	TCP	-	TLS/SSL

Les protocoles de messagerie s'appuient sur un mécanisme de publication et d'abonnement, où les transferts de données se font de manière asynchrone.

2.2.1.1 Hyper Text Transport Protocol (HTTP)

HTTP est le protocole fondamental du modèle client-serveur utilisé pour le Web, et le plus compatible avec l'infrastructure réseau existante. Actuellement, sa version la plus largement acceptée est HTTP/1.1. La figure ci-dessous nous montre un exemple de modèle d'interaction REST http.

Figure 2.1: Modèle d'interaction REST HTPP

Récemment, HTTP a été associé à REST [73], une directive pour le développement de services web basés sur un style architectural spécifique afin de définir l'interaction entre les différents composants. En raison du succès des services Web RESTful, beaucoup d'efforts ont été faits pour intégrer cette architecture aux systèmes basés sur l'IoT en combinant HTTP et REST.

❖ **Fonctionnement de REST HTTP**

La combinaison du protocole HTTP avec le protocole REST est louable, car les appareils peuvent rendre leurs informations d'état facilement accessibles, grâce à une méthode standardisée de création, de lecture, de mise à jour et de suppression des données (les opérations CRUD). Selon ce mappage, les opérations de création, de mise à jour, de lecture et de suppression des ressources correspondent respectivement aux méthodes HTTP POST, GET, PUT et DELETE. Pour les développeurs, le fait que REST établisse une cartographie de ces opérations CRUD avec des méthodes HTTP signifie qu'ils peuvent facilement construire un modèle REST pour différents dispositifs IoT [74]. La présentation des données n'est pas prédéfinie et en tant que telle, le type est arbitraire, les plus courants étant JSON et XML. Dans la plupart des cas, l'IoT se standardise autour de JSON par rapport à HTTP.

La figure 2.1 illustre un exemple d'interaction requête/réponse HTTP REST entre deux clients et un serveur.

❖ **Qualité de service et sécurité de REST HTTP**

HTTP utilise TCP comme protocole de transport. Bien que l'utilisation de TCP fournisse une livraison fiable de grandes quantités de données, ce qui est un avantage dans les connexions qui n'ont pas d'exigences strictes de latence, elle crée des défis dans les environnements à

ressources limitées [75]. L'un des principaux problèmes est que les nœuds contraints envoient la plupart du temps de petites quantités de données de manière sporadique et l'établissement d'une connexion TCP prend du temps et produit des frais généraux inutiles. Pour la QoS, HTTP ne fournit pas d'options supplémentaires, mais s'appuie sur TCP, qui garantit une livraison réussie tant que la connexion n'est pas interrompue.

HTTP en tant que mécanisme de sécurité utilise TLS [76] pour permettre un canal de communication crypté sécurisé, résultant en une version sécurisée de HTTP, également connu sous le nom de HTTPS. Les données sont cryptées pour empêcher quiconque d'écouter et de comprendre le contenu. Dans les systèmes qui comprendront des nœuds à ressources limitées, l'implémentation actuelle de TLS (TLS version 1.2) par son processus de handshake ajoute du trafic supplémentaire avec chaque établissement de connexion qui peut épuiser les capacités de calcul de ces dispositifs. Des efforts sont faits pour mettre au point une nouvelle version 1.3 de TLS qui rendra la poignée de main TLS plus rapide et plus légère, ce qui est plus pratique pour l'IoT [77] [78] [79].

HTTP ne définit pas explicitement les niveaux de QoS et nécessite un support supplémentaire pour cela. Cela a conduit à des modifications et à l'extension de HTTP, notamment sous la forme HTTP/2.0 [84], qui ont introduit un certain nombre d'améliorations, dont certaines sont particulièrement pertinentes dans le contexte de l'IoT. HTTP/2.0 permet une utilisation plus efficace des ressources réseau et une latence réduite en introduisant des en-têtes compressés, en utilisant un format de compression de mémoire très efficace et de faible capacité, ainsi que la possibilité d'utiliser plusieurs formats simultanés d'échanges sur le même lien [80].

Ces caractéristiques sont particulièrement intéressantes pour l'IoT car elles signifient que la taille des paquets est nettement plus petite, ce qui en fait une option plus adéquate pour les dispositifs à contrainte. De plus, il introduit ce que l'on appelle le push serveur, ce qui signifie que le serveur peut envoyer du contenu aux clients sans avoir besoin d'attendre leurs requêtes.

2.2.1.2 Constrained Application Protocol (CoAP)

Le protocole pour applications contraintes CoAP (Constrained Application Protocol) est un protocole qui spécifie le mode de fonctionnement des périphériques contraints de faible puissance dans l'Internet des objets (IoT, Internet of Things). Mis au point par l'IETF (Internet Engineering Task Force), CoAP est défini dans la spécification IETF RFC 7252 [81] [82].

CoAP est conçu pour permettre à des périphériques simples et contraints d'utiliser l'IoT, même par l'intermédiaire de réseaux contraints présentant peu de bande passante et de disponibilité. Le protocole sert généralement aux interactions de machine à machine (M2M).

Il est un protocole web basé sur une architecture client/serveur. Ce protocole reprend en partie les méthodes et nomenclatures du protocole HTTP. En revanche, contrairement au protocole HTTP, qui se base sur la suite TCP/IP, le protocole CoAP se base sur la suite UDP/IPv6/6LowPAN, dont les mécanismes d'échange de messages définis par le protocole UDP sont nettement allégés.

❖ **Fonctionnement de CoAP**

Le protocole CoAP utilise le protocole UDP au niveau de la couche transport, il supporte l'IPv6 pour la couche réseau et utilise le protocole de communication 802.15.4 au niveau de la couche liaison si l'on considère un réseau de capteurs. La taille des messages CoAP est allégée par rapport à celle des messages HTTP ; l'en-tête d'un message CoAP est fixé à 4 octets alors que celui des messages HTTP est variable. L'en-tête de chaque paquet contient le type de message et la qualité de service souhaitée pour la transmission du message :

- **Confirmable :** Message envoyé avec une demande d'accusé de réception, noté CON
- **Non-Confirmable :** Message envoyé sans demande d'accusé de réception, noté NON
- **Acknowledgment :** Accusé de réception du message de type « confirmable », noté ACK
- **Reset :** Accusé de réception d'un message qui n'est pas exploitable, noté RST

Pour transmettre une donnée (comme indiqué en exemple dans l'image ci-dessous), un client envoie à un serveur une requête CoAP, dans laquelle se trouve : le type du message (CON ou NON), l'identifiant du message (mid) et une action (GET, POST, PUT ou DELETE) sur une ressource identifiée par une URI [83]. Si la requête est du type CON alors le serveur retourne une réponse dans laquelle se trouve ; le type du message (ACK), le même mid que celui de la requête et un code réponse (2. xx, 4. xx ou 5.xx) et une représentation de la ressource.

Tout comme en HTTP, pour ses requêtes d'obtention de données du serveur, par exemple lors de l'obtention des valeurs du capteur, le client utilisera la méthode GET avec une URL de serveur, et comme réponse recevra un paquet avec ces données. La demande et les réponses sont appariées par un jeton ; un jeton dans la réponse doit être le même que celui défini dans la

demande. Il est également possible pour un client de transférer des données, par exemple des données de capteur mises à jour, vers un périphérique en utilisant la méthode POST à son URL. Cependant, CoAP a été conçu avec la fonctionnalité REST ; par conséquent, la conversion entre ces deux protocoles doit être mise en œuvre dans la chaîne de communication ; voir la figure 2.2.

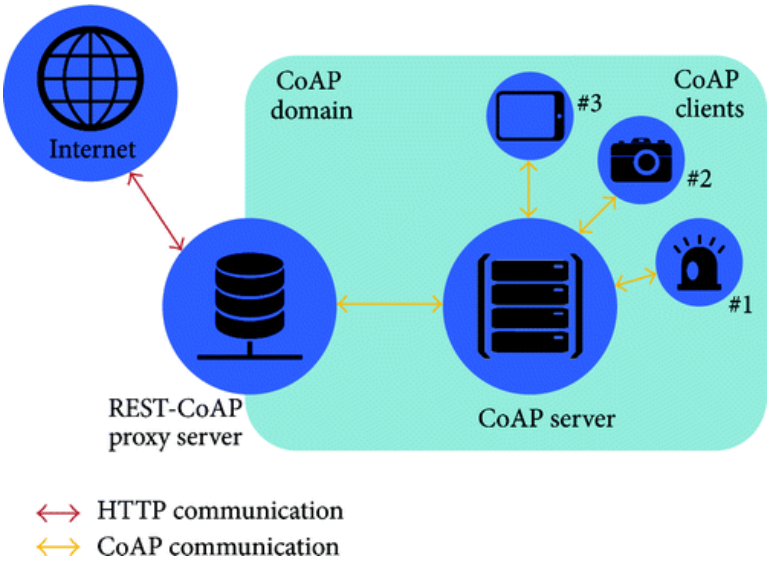

Figure 2.2:Fonctionnalité CoAP

CoAP a une fonction optionnelle qui peut améliorer le modèle requête/réponse en permettant aux clients de continuer à recevoir des changements sur une ressource demandée du serveur en ajoutant une option d'observation à une requête GET [84]. Avec cette option, le serveur ajoute le client à la liste des observateurs de la ressource spécifique, ce qui lui permettra de recevoir les notifications lorsque l'état de la ressource change. Afin de se rapprocher du paradigme publication/abonnement, l'IETF a récemment publié le projet de Publish-Subscribe Broker. Ce projet étend les capacités de la CoAP pour supporter les nœuds avec de longues interruptions de connectivité et/ou de temps de fonctionnement, avec des évaluations préliminaires des performances montrant des résultats prometteurs [85].

- **Sécurité de CoAP**

En tant que mécanisme de sécurité, CoAP utilise DTLS en plus de son protocole de transport UDP [86]. Il est basé sur le protocole TLS avec les modifications nécessaires pour faire fonctionner une connexion non fiable.

Comme DTLS n'a pas été conçu à l'origine pour l'IoT et les dispositifs à contrainte, de nouvelles versions optimisées pour les dispositifs légers sont apparues récemment [87] [88].

En raison de ses limites, l'optimisation de la DTLS pour l'IoT reste une question ouverte [89] [90].

2.2.1.3 Message Queue Telemetry Transport Protocol (MQTT)

MQTT est l'un des protocoles de messagerie légers qui suit le paradigme de l'abonnement à la publication, ce qui le rend plutôt adapté aux appareils à ressources limitées et aux conditions de connectivité réseau non idéales, comme une faible bande passante et une latence élevée. MQTT a été publié par IBM, avec sa dernière version MQTT v3.1 adoptée pour l'IoT par l'OASIS [70]. En raison de sa simplicité et d'un en-tête de message très petit par rapport aux autres protocoles de messagerie, il est souvent recommandé comme solution de communication de choix pour l'IoT. Ce protocole est principalement utilisé pour la remontée d'information d'objets connectés destinés aux particuliers. Quelques implémentations dans les objets connectés du monde industriel ont été réalisées

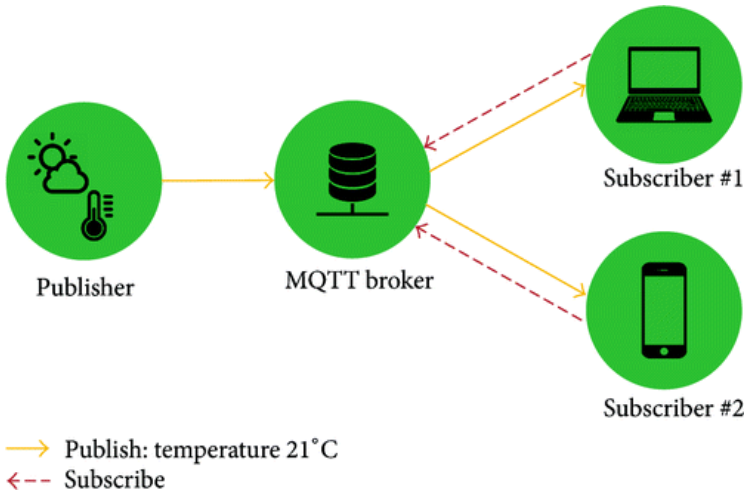

Figure 2.3: L'architecture du protocole MQTT

❖ **Fonctionnement de MQTT**

MQTT est un protocole de messagerie de publication et d'abonnement (publish/subscribe) basé sur le protocole TCP/IP. 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. Par la suite, ces messages peuvent être lus par des abonnés, appelés subscribers, qui au préalable ont établi une connexion de type 'subscribe' avec le broker. Ainsi, la transmission et la consommation des messages se font de manière asynchrone.

Figure 2.4: Modèle d'interaction de MQTT

Cependant, d'autres clients peuvent également s'abonner au même sujet et recevoir les mises à jour du courtier avec l'arrivée de nouveaux messages. Broker sert de composant central qui accepte les messages publiés par les clients et avec l'aide du sujet et le filtrage les livre aux clients abonnés. Dans le MQTT, nous pouvons utiliser un modèle d'interaction d'abonnement aux publications, comme illustré à la figure 2.4

Pour qu'un dispositif ait un rôle de courtier, il est nécessaire d'installer une bibliothèque de courtiers MQTT, par exemple Mosquitto broker, qui est l'un des courtiers MQTT open source les plus connus. Il convient de noter qu'il existe plusieurs autres courtiers de protocole du MQTT qui sont ouverts à l'utilisation et qui diffèrent par la mise en œuvre du protocole MQTT. Les clients sont disponibles en installant des bibliothèques clients MQTT. L'éditeur crée des sujets étiquetés dans le Courtier, comme le montre la figure. 2.4. Les sujets du MQTT sont traités comme une hiérarchie, avec des chaînes de caractères séparées par des barres obliques qui indiquent le niveau du sujet [91].

- **Format du message MQTT**

MQTT a la particularité d'être un protocole léger parce que le nombre de messages est restreint et ont des tailles faibles. En effet, chaque message se compose d'un en-tête fixe de 2 octets (spécifiant le type de message et le niveau de qualité de service employée), d'un en-tête variable facultatif, d'une payload (charge utile) limitée à 256 Mo. Le format de message du protocole MQTT est représenté à la figure 2.5. Les deux premiers octets font partie de l'en-tête fixe. En outre, le champ Message Type contient une variété de messages, par exemple, Connect (1), Connack (2), Publish (3) et Subscribe (8).

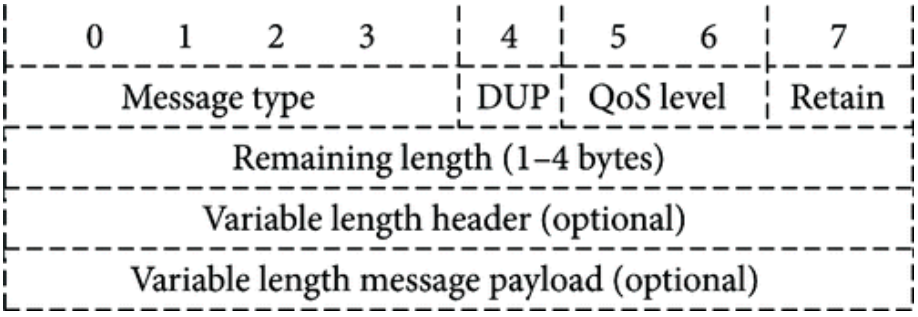

Figure 2.5: Format du messge MQTT

Ensuite, le drapeau DUP (en double) indique que le message est dupliqué et que le destinataire a déjà acquis ce message. Le champ QoS représente l'identification de trois niveaux de QoS pour la livraison des messages de publication. Le champ suivant est appelé Conserver et informe le serveur de conserver le dernier message de publication et de soumettre ce message aux nouveaux abonnés (ce message sera envoyé en tant que premier message). Le dernier champ (champ restant) indique la longueur restante du message (c.-à-d. Pièces optionnelles).

- **Qualité de services et Sécurité dans MQTT**

Il existe trois niveaux de qualité de service (QoS : QoS, QoS 0, 1 et 2) qui déterminent la façon dont le protocole MQTT gère le contenu [92] [93]. Les clients abonnés peuvent spécifier le niveau de QoS maximal qu'ils souhaitent recevoir. Toutefois, plus le niveau de qualité est élevé et plus cela est gourmand en termes de latence et de bande passante, à cause des répétitions et des accusés de réception supplémentaires.

Le choix du niveau peut être défini à la fois dans la publication et dans le corps du message d'abonnement. La QoS 0 est livrée sur la base du meilleur effort, sans confirmation à la réception du message. Il s'agit d'un choix dans les cas où certains capteurs recueillent des informations de télémétrie sur une plus longue période de temps et où les valeurs des capteurs ne changent pas de façon significative.

Le niveau de garantie suivant est la QoS 1, qui assure que les messages arriveront, donc une confirmation du message est nécessaire. Cela signifie que le destinataire doit envoyer un accusé de réception, et s'il n'arrive pas dans un délai défini, l'éditeur enverra à nouveau un message de publication. La troisième option, la QoS 2, garantit que le message sera délivré exactement une fois sans duplication. La quantité de ressources nécessaires pour traiter les paquets MQTT augmente avec le niveau de QoS choisi, il est donc important d'ajuster le choix de QoS aux conditions spécifiques du réseau.

MQTT utilise TCP, ce qui peut être critique pour les appareils à contrainte. A cette fin, une solution a été proposée en tant que version MQTT pour les réseaux de capteurs (MQTT-SN) qui utilise UDP et supporte l'indexation des noms de sujets [94].

Cette solution ne dépend pas de TCP, mais utilise UDP comme option de transport plus rapide, plus simple et plus efficace sur une liaison sans fil [95]. L'autre caractéristique importante améliorée est la taille réduite des charges utiles. Ceci est fait en numérotant les paquets de données avec des identifiants de sujet numériques plutôt qu'avec des noms de sujet longs. Le plus grand inconvénient est qu'actuellement MQTT-SN n'est supporté que par quelques plates-formes, et qu'il n'y a qu'une seule implémentation de courtier libre connue, appelée petit Agent de messages [96].

Comme il a été conçu pour être aussi léger, le MQTT ne fournit pas de cryptage et, au lieu de cela, les données sont échangées en texte clair, ce qui est clairement un problème du point de vue de la sécurité. Par conséquent, le cryptage doit être implémenté en tant que fonctionnalité séparée, par exemple via TLS. L'authentification est mise en œuvre par de nombreux courtiers MQTT, par l'intermédiaire de l'un des paquets de messages de type contrôle MQTTs, appelé CONNECT. Les courtiers exigent des clients qu'en envoyant le message CONNECT, ils définissent la combinaison nom d'utilisateur/mot de passe avant de valider la connexion ou de la refuser si l'authentification n'a pas été réussie. Dans l'ensemble, la sécurité est un effort continu pour le MQTT [97], et probablement le plus important puisque le MQTT est l'une des solutions de protocole de communication les plus largement adoptées et les plus avancées. Résoudre le problème de la sécurité créerait un avantage important pour le MQTT par rapport à d'autres instituts des solutions. MQTT-S / MQTT-SN est une extension de MQTT, qui est conçue pour les appareils à faible consommation et à faible coût. Il est basé sur MQTT mais présente quelques optimisations pour les WSN. Les noms de rubrique sont remplacés par des ID de rubrique, ce qui réduit les frais généraux de transmission. Les sujets n’ont pas besoin d’être enregistrés car ils sont pré-enregistrés.

2.2.1.4 Data Distribution Service (DDS)

DDS est une norme d'interopérabilité centrée sur les données en temps réel qui utilise un modèle d'interaction publication-abonnement, tel que défini par le groupe de gestion des objets [98]. Contrairement à d'autres protocoles de publication avec abonnement, DDS est décentralisé et basé sur une communication entre homologues. Dans DDS, les éditeurs et les abonnés peuvent communiquer en tant qu'homologues via le bus de données, permettant ainsi un

échange de données asynchrone basé sur leurs intérêts. Le fait qu'il n'y ait pas de courtier diminue également la probabilité de défaillance du système, car il n'existe pas de point de défaillance unique pour l'ensemble du système, ce qui rend un système plus fiable. Les deux côtés de la communication sont découplés l'un de l'autre et un éditeur peut publier des données même s'il n'y a aucun abonné intéressé. L'utilisation des données est fondamentalement anonyme, car les éditeurs ne demandent pas qui consomme leurs données.

L'une des principales caractéristiques du protocole DDS est son évolutivité, qui découle de sa prise en charge de la découverte dynamique. Le processus de découverte, réalisé via le protocole de découverte intégré DDS, permet aux abonnés de savoir quels éditeurs sont présents, de spécifier les informations qui les intéressent avec la qualité de service souhaitée, et aux éditeurs de publier leurs données [99].

DDS garantit que les nœuds appropriés de publication et d'abonnement seront connectés et que l'échange de données se fera en temps réel. Une autre caractéristique importante et unique du DDS est son orientation vers les données, contrairement à la plupart des protocoles qui sont centrés sur les messages. Pour le paradigme centré sur les données, ce qui compte le plus, ce sont les données auxquelles les clients veulent accéder. Par conséquent, l'accent est mis sur les informations de contenu elles-mêmes. Ainsi, DDS permet une architecture dans laquelle les nœuds participants comprennent la valeur de données de manière cohérente. Dans DDS, le type et le contenu des données définissent la communication, tandis que dans les protocoles centrés sur les messages, l'accent est mis sur les opérations et les mécanismes permettant de fournir ces données. L'approche centrée sur les données de DDS peut être utilisée lorsque les architectes système définissent les "sujets" en regroupant des éléments de données pouvant être liés logiquement dans le but de garantir une meilleure évolutivité et de meilleurs résultats.

- **Entités et Fonctionnement du DDS**

Les principales entités de l'architecture DDS sont les suivantes : domaine, participant au domaine, sujet, éditeur, abonné, DataWriter et lecteur de données [100]. Les éditeurs et les abonnés sont divisés en domaines, une entité de concept virtuel qui permet d'isoler la communication au sein de nœuds ayant des intérêts communs. Le participant au domaine est le point d'entrée pour l'échange de messages dans des domaines spécifiques. Associe les éditeurs et les abonnés et les domaines auxquels ils appartiennent. Il est utilisé pour créer des éditeurs, des abonnés, des rédacteurs de données, des lecteurs de données et des sujets au sein d'un domaine.

L'architecture de DDS définit deux couches : (i) Data-Centric Publish-Subscribe (DCPS) et (ii) Data-Local Reconstruction Layer (DLRL). La couche DCPS est responsable de la livraison des informations aux destinations finales (abonnés). DLRL représente, d'autre part, une couche facultative qui sert de pont / interface aux fonctionnalités de DCPS (partage de données distribuées entre objets distribués) [101]. Dans la couche DCPS, cinq entités gèrent le flux de données (voir la figure 2.6) :

i- Éditeur : Il envoie les ensembles de données requis ;

ii- DataWriter : Il est utilisé par l'application pour interagir avec l'éditeur en ce qui concerne les valeurs et les modifications concernant le type de données. La coopération étroite entre Publisher et DataWriter est utilisée par l'application pour la publication de données dans le contexte fourni ;

iii- Subscriber : Il reçoit les données de l'éditeur et les transfère à l'application ;

iv- DataReader : Il est contrôlé par l'abonné de lire les données reçues ;

v- Sujet : Il est défini par le type et le nom des données et relie les DataWriters aux DataReaders.

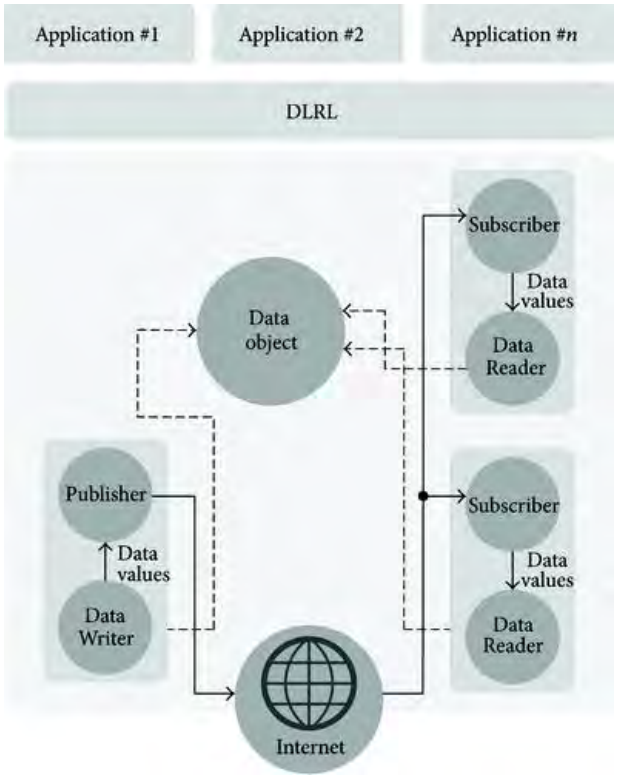

Figure 2.6: Architecture Conceptuelle du DDS

Le middleware de mise en œuvre DDS, définit la manière dont les données sont structurées, modifiées et accessibles dans un espace de données abstrait [102]. Pour ce faire, nous utilisons

une abstraction d'espace de données où tous les clients peuvent accéder pour lire ou stocker leurs données, appelée Global Data Space (GDS). C'est dans les GDS que la fonctionnalité de découverte dynamique DDS entre en jeu en permettant aux éditeurs et aux abonnés qui rejoignent le GDS de découvrir automatiquement leur existence mutuelle ainsi que leurs intérêts.

L'unité d'échange d'informations entre les nœuds DDS dans GDS est une rubrique et est définie par un nom, un type de données et un ensemble de stratégies de qualité de service. Les éditeurs et les abonnés sont les entités de distribution et de consommation des données, qui publient et reçoivent des données via le répertoire de stockage global, mais ils ne peuvent le faire eux-mêmes. Au lieu de cela, les éditeurs utilisent Data Writers pour envoyer des données et les abonnés utilisent des lecteurs de données pour recevoir des données [103] avec la correspondance entre les deux sujets, c'est-à-dire que pour communiquer entre eux, les éditeurs et les abonnés doivent utiliser le même sujet (même nom)., tapez et une QoS compatible).

❖ **Sécurité de DDS**

DDS utilise UDP par défaut, mais il peut également prendre en charge TCP. Un autre protocole important dans DDS est le protocole filaire RTPS (Real Time Publish Subscribe) [104], qui représente le protocole d'interopérabilité DDS permettant le partage de données entre différentes implémentations de fournisseurs. L'un des avantages de l'utilisation de DDS est un large ensemble de politiques de qualité de service (plus de 20 valeurs de service définies dans la norme). Lors de l'envoi de données, les stratégies de qualité de service de chaque sujet, les rédacteurs de données et les éditeurs contrôlent le moment et celui d'envoi des données au middleware. D'autre part, la rubrique QoS, les lecteurs de données et les abonnés contrôlent le comportement lors de la réception des données. Ces différentes stratégies gèrent une multitude de fonctionnalités DDS, telles que la découverte d'entités distantes distribuées, la fourniture de données, la disponibilité, l'heure et l'utilisation des ressources [105].

Pour un mécanisme de sécurité, DDS implémente diverses solutions. Sur la base du protocole de transport choisi, TLS peut être utilisé dans le cas où TCP est le protocole de transport, ou le protocole DTLS dans le cas où UDP est utilisé. De même pour TLS, DTLS génère également des frais généraux excessifs dans des environnements soumis à des contraintes, pour lesquels des mécanismes améliorés ont été proposés. À cette fin, la spécification de sécurité DDS OMG (groupe de gestion d'objets) définit une architecture étendue de modèle de sécurité et d'interface de plug-in de service (SPI) conçue pour les

implémentations DDS adaptées aux systèmes IoT [106]. La question de la spécification de la sécurité est encore ouverte pour DDS et de nouveaux ajouts devraient être mis en place dans le futur. L'un des ajouts attendus est un mécanisme de découverte sécurisée capable d'établir des flux de transport sécurisé entre applications basées sur DDS ayant une classification de sécurité correspondante, comme proposé dans [107].

DDS est une solution importante pour les environnements IoT pour son architecture de publication / abonnement décentralisée et sa prise en charge pour la mise en œuvre à la fois dans des périphériques puissants et des périphériques sous contraintes [108]. Le défi de DDS réside dans le fait qu'il n'a pas été largement utilisé, bien que cela puisse changer avec les nouvelles implémentations de DDS open source prêtes à être testées, telles que OpenDDS [109].

2.2.1.5 Advanced Message Queueing Protocol (AMQP)

AMQP est un protocole standard ouvert qui suit le paradigme publication-souscription défini par OASIS [110], conçu pour permettre l'interopérabilité entre un grand nombre d'applications et de systèmes différents, quelles que soient leurs conceptions internes. Cette fonctionnalité d'interopérabilité de AMQP est importante dans la mesure où elle permet à différentes plates-formes d'échanger des messages, ce qui peut être particulièrement utile dans les systèmes hétérogènes [111] [112]. Ce protocole convient mieux aux parties du système où la bande passante et le temps de latence ne sont pas limités, avec plus de puissance de traitement.

❖ **Fonctionnement de AMQP**

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. La communication via l'AMQP est réalisée par deux composants clés [113] :

i- Échanges : Ils sont utilisés pour acheminer les messages vers les files d'attente appropriées : Le routage (entre échanges et files d'attente) repose sur des règles / exigences prédéfinies ;

ii- Message Queues : Ils sont stockés dans des files d'attente de messages avant d'envoyer à la destination finale (récepteur) ;

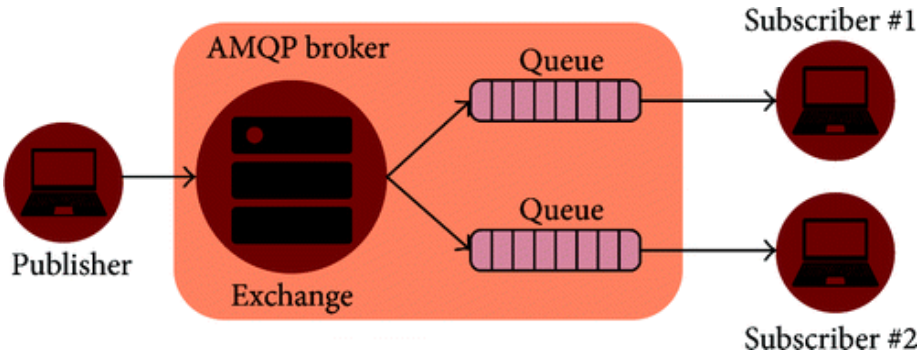

Figure 2.7: Mécanisme de publication / abonnement AMQP

❖ **Format de Message AMQP**

AMQP définit la couche de messagerie en haut de la couche de transport (TCP est utilisé comme protocole de transport) où toutes les capacités de messagerie sont traitées. Ensuite, deux types de messages sont définis dans AMQP : (i) messages nus (du côté de l'expéditeur) et (ii) messages annotés (du côté du destinataire); voir la figure 2.8.

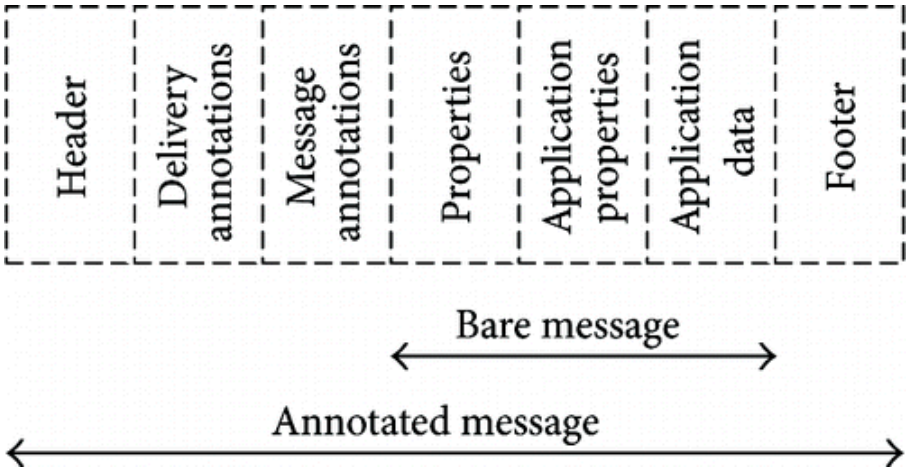

Figure 2.8: Formation de message AMQP

L'en-tête permet une livraison de paramètres incluant la durabilité, la priorité, le temps de vie (TTL), le premier acquéreur et le nombre de livraison.

Dans le contexte IoT, AMQP est le plus approprié pour le plan de contrôle ou les fonctions d'analyse basées sur le serveur. Par conséquent, ce n'est pas un candidat approprié pour la transmission de données M2M (communication E2E).

2.2.1.6 Prototocole XMPP

XMPP est un protocole de messagerie standard ouvert formalisé par l'IETF [114]. Il a été initialement conçu pour la messagerie instantanée et l'échange de messages entre les applications. Il s'agit d'un protocole textuel basé sur XML (Extensible Markup Language) qui

met en œuvre une interaction client-serveur et une interaction publication-abonnement [115], s'exécutant sur le protocole TCP. Dans les solutions IoT, il est conçu pour permettre aux utilisateurs d'envoyer des messages en temps réel, en plus de gérer la présence de l'utilisateur. XMPP permet aux applications de messagerie instantanée de bénéficier de toutes les fonctionnalités de base, notamment l'authentification, le cryptage de bout en bout et la compatibilité avec d'autres protocoles [116].

La fonctionnalité globale du protocole XMPP est représentée sur la figure 2.5 où la passerelle peut résoudre les problèmes liés à l'envoi de messages entre des réseaux étrangers ; XMPP connecte des clients et des serveurs en utilisant la stance appelée XML [117].

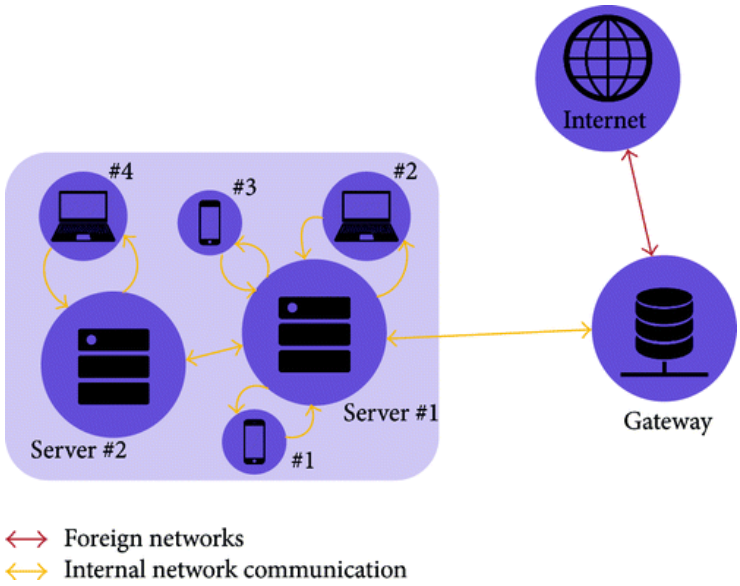

Figure 2.9: Communication dans XMPP

XMPP (Extensible Messaging and Presence Protocol) est un protocole ouvert de messagerie instantanée. Le protocole de messagerie instantanée Jabber est notamment basé dessus. Le procotole est défini par plusieurs RFC (RFC6120, 6121, 6122, 3922 et 3223) qui définissent, à la base, les infrastructures Jabber.

Aujourd'hui, le protocole XMPP n'est plus seulement utilisé pour de la messagerie instantanée. Une implémentation de XMPP a été développée pour l'IoT (XMPP-IoT). Plusieurs API existent dans divers langages.

❖ **Fonctionnement de XMPP**

XMPP prend en charge le modèle d'interaction client-serveur, mais de nouvelles extensions permettent également l'utilisation du modèle de publication / abonnement générique. Ces extensions permettent aux entités XMPP de créer des rubriques et de publier des informations.

Une notification d'événement est ensuite diffusée à toutes les entités qui se sont abonnées à un nœud spécifique. Cette fonctionnalité est plutôt importante pour les scénarios IoT-fog-cloud, car elle constitue le fondement d'une grande variété d'applications nécessitant des notifications d'événement. Les clients et les serveurs de XMPP communiquent entre eux via des flux XML pour échanger des données sous la forme de strophes XML (unités de données structurées sémantiques) [14]. Trois types de strophes sont définis :

<presence />, <message /> et <iq /> (info / requête). Une strophe de messages définit un titre et un contenu de message et est utilisée pour envoyer des données entre entités XMPP. Les strophes de messages ne reçoivent pas d'accusé de réception de la part de l'entité réceptrice, qu'il s'agisse du client ou du serveur. Une strophe de présence affiche et notifie aux entités les mises à jour de statut ayant le rôle d'abonnement dans XMPP.

S'il existe un intérêt pour la présence de certains JID (Jaber ID - une adresse de nœud dans XMPP), un client s'abonne à leur présence et chaque fois qu'un nœud envoie une mise à jour de présence, un client est notifié. Une strophe iq associe expéditeurs et destinataires de messages. Il est utilisé pour obtenir des informations du serveur, par exemple des informations sur le serveur ou ses clients enregistrés, ou pour appliquer certains paramètres au serveur. Sa fonction est similaire aux méthodes HTTP GET et POST.

L'une des caractéristiques les plus importantes de ce protocole réside dans ses fonctions de sécurité, ce qui en fait l'un des protocoles de messagerie les plus sécurisés étudiés. Contrairement aux autres protocoles étudiés, par exemple MQTT et CoAP, dans lesquels les cryptages TLS et DTLS ne sont pas intégrés aux spécifications de protocole, la spécification XMPP intègre déjà des mécanismes TLS, qui fournissent un mécanisme fiable pour assurer la confidentialité et l'intégrité des données. Les nouveaux ajouts aux spécifications XMPP incluent également des extensions liées à la sécurité, à l'authentification, à la confidentialité et au contrôle d'accès. Outre TLS, XMPP implémente SASL, qui garantit la validation du serveur via un profil spécifique à XMPP [118].

Depuis que XMPP a été initialement conçu pour la messagerie instantanée, il existe une faiblesse potentielle notable. En utilisant XML, la taille des messages le rend peu pratique dans les réseaux soumis à des contraintes de bande passante. L'absence de garanties de qualité de service fiables constitue un autre inconvénient. Etant donné que XMPP s’exécute sur une connexion TCP persistante et qu’il n’a pas de codage binaire efficace, il n’a pas été pratique de l’utiliser sur des réseaux sans fil à faible consommation avec pertes, souvent associés aux

technologies IoT. Cependant, ces derniers temps, de nombreux efforts ont été déployés pour rendre XMPP mieux adapté à l'IoT [119] [120] [121]. Dans [122], un schéma de publication / abonnement XMPP léger a été présenté pour les périphériques IoT limités en ressources, améliorant et optimisant ainsi la version existante du même protocole.

❖ **Fonctionnement et type de message**

Les fonctions intégrées rendent XMPP un protocole préféré par la plupart des applications de messagerie instantanée et donc pertinent pour une partie spécifique de l'écosystème IoT. XMPP implémente également un bloc de construction pour une communication sécurisée et permet de nouvelles applications en plus des protocoles de base (Extensible Messaging and Presence Protocol (XMPP) [123]. Comme on l'a mentionné, XMPP utilise les soi-disant strophes qui divisent le code en trois composantes : (i) message, (ii) présence et (iii) info / requête ; voir la figure 2.6.

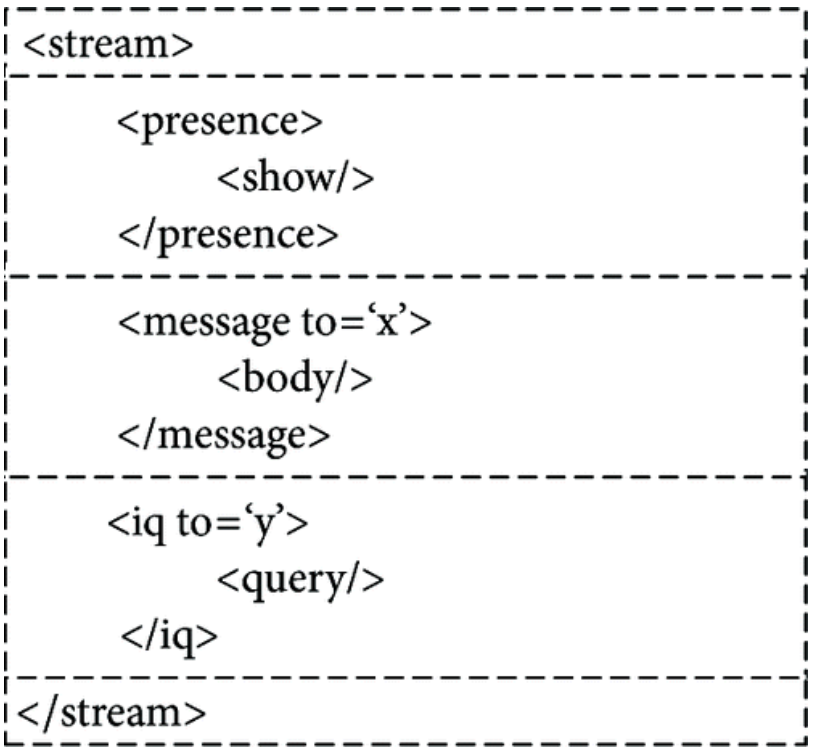

Figure 2.10: Structure de strophe XMPP

Les messages en strophe identifient l'adresse source, l'adresse de destination, les types et les ID des entités XMPP qui fournissent la méthode PUSH pour la récupération des données. La strophe de présence informe les utilisateurs finaux des mises à jour d'état. Enfin, la stase iq fait l'association entre les expéditeurs de messages et les récepteurs. L'inconvénient possible de XMPP est la communication par texte utilisant XML. Cela entraîne une charge de réseau plus

élevée (frais généraux). Par conséquent, il existe une solution possible à ce problème : flux XML utilisant EXI [124] [125].

XMPP est très souvent comparé au protocole SIP où SIP est intrinsèquement un protocole peer-to-peer alors que XMPP est intrinsèquement client-serveur. Les tâches qui sont faciles dans les systèmes client-serveur, par exemple, pour partager l'état, pour enregistrer les données sur le serveur ou pour afficher des messages hors ligne sur le serveur, sont bien réalisées avec le protocole XMPP. D'autre part, l'un des principaux objectifs de SIP (décrit plus loin dans cette section) est de garder l'intelligence au point final. Idéalement, un serveur proxy SIP ne maintient même pas l'état de session pour la boîte de dialogue SIP (RFC 6120) [126].

2.2.1.7 Comparaison de quelques des protocoles

Sur la base des faits susmentionnés, on peut voir que MQTT, XMPP ou CoAP peuvent être utilisés comme protocoles pour les données M2M, mais il existe encore des limitations (par exemple, la durée du message et l'inactivité des capteurs) principalement associés à la connectivité de bout en bout. Dans la littérature, on peut trouver plusieurs comparaisons entre ces protocoles. Par exemple, [127], les auteurs comparent le délai de transmission de bout en bout et l'utilisation de la bande passante de MQTT et de CoAP ; Le MQTT fournit les données requises avec un retard inférieur par rapport à CoAP en cas de perte de paquets faible. Sinon, en cas de perte élevée de paquets, CoAP a donné de meilleurs résultats. Dans une autre recherche [128], l'attention a porté sur l'environnement d'application des smartphones. Les résultats montrent que CoAP offre une utilisation de bande passante inférieure et un temps de trajet aller-retour inférieur (RTT) que MQTT.

En proposant un nouveau point de vue dans cet espace, dans notre projet [129], nous proposons un nouveau moyen de transmettre les données M2M via le réseau en ce qui concerne la longueur des données transmises, la programmation des transmissions et la nature E2E de la communication M2M. Le protocole SIP est une partie courante des réseaux cellulaires d'aujourd'hui et même s'il a été inventé pour différentes applications comme la téléphonie IP, la diffusion multimédia et la messagerie instantanée, c'est un excellent candidat pour devenir un bus de communication principal pour les composantes constitutives de l'écosystème IoT (dans le cadre de la vision émergente 5G).

Tableau 2.3: Comparaison des protocole d'application de l'IoT

Protocols	RESTful	Transport	Publish/subscribe	Request/response	Security	QoS	Header size (bytes)
CoAP	√	UDP, SMS	√	√	DTLS	√	4
MQTT	×	TCP	√	×	SSL	√	2
MQTT-SN	×	TCP	√	×	SSL	√	2
XMPP	×	TCP	√	√	SSL	×	—
AMQP	×	TCP	√	×	SSL	√	8
SIP	×	TCP, UDP, SMS	√	√	SSL, TLS	√	—
DDS	×	TCP, UDP	√	×	SSL, DTLS	×	—
HTTP	√	TCP	×	√	SSL	×	—

2.2.2 Protocoles de découvertes de services

La grande évolutivité de l'IoT nécessite un mécanisme de gestion des ressources capable d'enregistrer et de découvrir des ressources et des services de manière auto-configurée, efficace et dynamique. Les protocoles les plus dominants dans ce domaine sont les DNS multicast (mDNS) et DNS Service Discovery (DNS-SD) qui peuvent découvrir les ressources et les services offerts par les appareils IoT. Bien que ces deux protocoles aient été conçus à l'origine pour des dispositifs riches en ressources, il existe des études de recherche qui en adaptent des versions allégées aux environnements IoT.

2.2.2.1 Multicast DNS (mDNS)

La résolution de noms est un service de base pour certaines applications IoT telles que le chat. mDNS est un tel service qui peut effectuer la tâche de serveur DNS monodiffusion. mDNS est flexible du fait que l'espace de noms DNS est utilisé localement sans frais ni configuration supplémentaires. mDNS est un choix approprié pour les appareils embarqués basés sur Internet car a) Il n'y a pas besoin de reconfiguration manuelle ou d'administration supplémentaire pour gérer les appareils ; b) Il est capable de fonctionner sans infrastructure ; et c) il est capable de continuer à fonctionner en cas de défaillance de l'infrastructure.

mDNS demande les noms en envoyant un message IP multicast à tous les nœuds du domaine local. Par cette requête, le client demande aux appareils qui ont le nom donné de répondre. Lorsque la machine cible reçoit son nom, elle multidiffuse un message de réponse contenant son adresse IP. Tous les périphériques du réseau qui obtiennent le message de réponse mettent à jour leur cache local à l'aide du nom et de l'adresse IP donnés.

2.2.2.2 DNS Service Discovery (DNS-SD)

La fonction d'appariement des services requis par les clients utilisant mDNS est appelée découverte de services DNS (DNS-SD). À l'aide de ce protocole, les clients peuvent découvrir un ensemble de services souhaités dans un réseau spécifique en utilisant des messages DNS

standard. La figure 18 fournit une illustration visuelle du fonctionnement de ce protocole. DNS-SD, comme mDNS, fait partie des aides à la configuration zéro pour connecter des machines sans administration ou configuration externe.

Essentiellement, DNS-SD utilise mDNS pour envoyer des paquets DNS à des adresses multicast spécifiques via UDP. Il existe deux étapes principales pour traiter la découverte de services : rechercher les noms d'hôte des services requis tels que les imprimantes et coupler les adresses IP avec leurs noms d'hôte à l'aide de mDNS. La recherche de noms d'hôtes est importante, car les adresses IP peuvent changer, contrairement aux noms. La fonction de couplage diffuse les détails des pièces jointes réseau comme l'IP et le numéro de port vers chaque hôte associé. En utilisant DNS-SD, les noms d'instance du réseau peuvent être maintenus constants aussi longtemps que possible pour augmenter la confiance et la fiabilité. Par exemple, si certains clients connaissent et utilisent une imprimante spécifique aujourd'hui, ils pourront l'utiliser par la suite sans aucun problème.

2.2.3 Protocoles d’infrastructures

2.2.3.1 RPL

Le protocole de routage pour les réseaux à faible puissance et avec perte (RPL) est un protocole à vecteur de distance qui peut prendre en charge une variété de protocoles de liaison de données, y compris ceux discutés dans la section précédente. Il construit un graphe acyclique dirigé orienté destination (DODAG) qui n'a qu'une seule route de chaque nœud feuille à la racine vers laquelle tout le trafic du nœud sera acheminé. Dans un premier temps, chaque nœud envoie un DODAG Information Object (DIO) s'annonçant comme racine. Ce message se propage dans le réseau et l'ensemble du DODAG est progressivement construit. Lors de la communication, le nœud envoie un objet d'annonce de destination (DAO) à ses parents, le DAO est propagé à la racine et la racine décide où l'envoyer en fonction de la destination. Lorsqu'un nouveau nœud souhaite rejoindre le réseau, il envoie une demande de sollicitation d'informations DODAG (DIS) pour rejoindre le réseau et la racine répondra avec un accusé de réception DAO (DAO-ACK) confirmant la jointure. Les nœuds RPL peuvent être sans état, ce qui est le plus courant, ou avec état. Un nœud sans état conserve uniquement les traces de ses parents. Seul root a la connaissance complète de l'ensemble du DODAG. Par conséquent, toutes les communications passent par la racine dans tous les cas. Un nœud avec état garde la trace de ses enfants et de ses parents et, par conséquent, lorsqu'il communique à l'intérieur d'un sous-arbre du DODAG, il n'a pas besoin de passer par la racine [RFC6550].

2.2.3.2 6LoWPAN

Le réseau personnel sans fil IPv6 sur faible puissance (6LoWPAN) est le premier standard et le plus couramment utilisé dans cette catégorie. Il encapsule efficacement les en-têtes longs IPv6 dans les petits paquets IEEE802.15.4, qui ne peuvent pas dépasser 128 octets. La spécification prend en charge différentes adresses de longueur, une faible bande passante, différentes topologies, y compris en étoile ou maillage, la consommation d'énergie, des réseaux à faible coût et évolutifs, la mobilité, le manque de fiabilité et une longue durée de sommeil. La norme fournit la compression d'en-tête pour réduire la surcharge de transmission, la fragmentation pour atteindre la longueur de trame maximale de 128 octets dans IEEE802.15.4 et la prise en charge de la livraison multi-sauts. Les cadres dans 6LoWPAN utilisent quatre types d'en-têtes : pas d'en-tête 6loWPAN (00), d'en-tête de répartition (01), d'en-tête de maillage (10) et d'en-tête de fragmentation (11). Dans le cas d'en-tête No 6loWPAN, toute trame qui ne suit pas les spécifications 6loWPAN est rejetée. L'en-tête de répartition est utilisé pour la multidiffusion et les compressions d'en-tête IPv6. Les en-têtes maillés sont utilisés pour la diffusion; tandis que les en-têtes de fragmentation sont utilisés pour briser un long en-tête IPv6 pour s'adapter à des fragments d'une longueur maximale de 128 octets.

2.2.3.3 LTE-A

Long-Term Evolution Advanced (LTE-A) est un ensemble de normes conçues pour s'adapter aux applications de communication M2M et IoT dans les réseaux cellulaires. LTE-A est un protocole évolutif et moins coûteux par rapport aux autres protocoles cellulaires. LTE-A utilise OFDMA (Orthogonal Frequency Division Multiple Access) comme technologie d'accès à la couche MAC, qui divise la fréquence en plusieurs bandes et chacune peut être utilisée séparément. L'architecture de LTE-A se compose d'un réseau central (CN), d'un réseau d'accès radio (RAN) et des nœuds mobiles. Le CN est responsable du contrôle des appareils mobiles et du suivi de leurs adresses IP. RAN est responsable de l'établissement des plans de contrôle et de données et de la gestion de la connectivité sans fil et du contrôle d'accès radio. RAN et CN communiquent en utilisant la liaison S1, comme illustré à la Figure 4où RAN se compose des eNB auxquels d'autres nœuds mobiles sont connectés sans fil.

Il existe d’autres protocoles d’infrastructures comme z-wave, EPCglobal, etc. sur le tableau suivant nous avons ressencé les protocoles les plus utilisés.

Tableau 2.4:Protocoles Standarisés de l'IoT

<table>
<tr><td colspan="2">Application Protocol</td><td>DDS</td><td>CoAP</td><td>AMQP</td><td>MQTT</td><td>MQTT-SN</td><td>XMPP</td><td>HTTP REST</td></tr>
<tr><td colspan="2">Service Discovery</td><td colspan="4">mDNS</td><td colspan="3">DNS-SD</td></tr>
<tr><td rowspan="4">Infrastructure Protocols</td><td>Routing Protocol</td><td colspan="7">RPL</td></tr>
<tr><td>Network Layer</td><td colspan="5">6LoWPAN</td><td colspan="2">IPv4/IPv6</td></tr>
<tr><td>Link Layer</td><td colspan="7">IEEE 802.15.4</td></tr>
<tr><td>Physical/Device Layer</td><td>LTE-A</td><td colspan="2">EPCglobal</td><td colspan="2">IEEE 802.15.4</td><td colspan="2">Z-wave</td></tr>
<tr><td colspan="2">Influential Protocol</td><td colspan="4">IEEE 1888.3, IPSec</td><td colspan="3">IEEE 1905.1</td></tr>
</table>

2.2.4 Protocole SIP

Le protocole SIP (Session Initiation Protocol) est un protocole de signalisation défini par l'IETF (Internet Engineering Task Force) [130] permettant d'établir, libérer et modifier des sessions multimédias (RFC 3261). Il s'appuie sur le modèle client/serveur comme les protocoles classiques tels que HTTP (Hyper Text Transport Protocol) et SMTP (Simple Mail Transport Protocol). Il utilise un système d'adressage basé sur des URL (Uniform Resource Locator) similaires au serveur de messagerie.

Aussi, il dispose des extensions supportant de nombreux services tels que la présence, le transfert d'appel, la conférence, les services additionnels de la téléphonie classique et la messagerie instantanée. Des messages courts non relatifs à un appel peuvent toutefois être transportés par SIP de la même manière qu'un SMS classique. Samuel OUYA et al. [131] ont proposé une approche de SMS basé sur le protocole SIP.

2.2.4.1 Transactions SIP

Avant l'initialisation d'une session SIP, l'utilisateur est localisé sur le réseau avec une adresse URI (Uniform Resource Identifier). Les terminaux impliqués dans une session SIP doivent donc s'identifier par cet URI qui définit une syntaxe permettant de désigner de manière unique, formelle et normalisée une ressource, qu'il s'agisse d'un document textuel, audio ou vidéo. Le format d'un URI se présente de la manière suivante :

sip : identifiant [: pwd]@adresse_serveur [?paramètre]

- **Le mot-clé sip** spécifie le protocole à utiliser pour la communication ;

- **La partie identifiant** définit le nom ou le numéro de l'utilisateur ;

- La partie pwd est facultative. Elle est obligatoire lorsqu'on veut s'authentifier auprès d'un serveur ;

- **La partie adresse_serveur** spécifie le serveur chargé du compte SIP dont l'identifiant précède l'arobase. Le serveur est spécifié par son adresse IP ou par un nom qui sera résolu par le DNS (Domaine Name System) ;

- **La partie paramètre est facultative**. Les paramètres permettent soit de modifier le comportement par défaut (par exemple, en modifiant les protocoles de transport ou les ports, ou encore la durée de vie par défaut d'une requête).

En outre, le protocole SIP utilise les protocoles TCP (Transmission Control Protocol) et UDP (User Datagram Protocol). Pour acheminer des requêtes de signalisation. Toutefois, si aucun protocole de transport n'est précisé dans l'URI de la requête, le protocole UDP est utilisé par défaut.

2.2.4.2 Méthodes et réponse SIP

Pendant l'initialisation d'une session SIP entre deux terminaux, les méthodes suivantes sont échangées lors de la mise en relation de l'appelant et de l'appelé :

- INVITE : Invite le terminal SIP d'un utilisateur à participer à une session.

- ACK : Confirme que l'appelant a reçu une réponse à sa requête INVITE.

- BYE : Met fin à une connexion entre utilisateurs ou lorsque le terminal refuse l'invitation à participer à une session.

- CANCEL : Annule une requête.

- OPTIONS : Sollicite du Proxy Server afin qu'il précise ses capacités à contacter le terminal de l'appelé.

- REGISTER : C'est une méthode utilisée par le client pour enregistrer l'adresse listée dans paramètre TO de l'URL par le serveur auquel il est relié. Les requêtes sont traitées par le client de manière ordonnée pour éviter l'envoyer de nouvelle requête REGISTER tant qu'il n'aura pas traité la précédente. Le client doit définir une adresse d'enregistrement du type utilisateur@domaine. Cette méthode assure également un service de localisation.

Le type des réponses aux requêtes envoyées dans les transactions SIP sont décrites dans le tableau ci-dessous.

Tableau 2.5: Code d'état

Code	Description
1XX	Les réponses Informational (1xx) sont utilisées pour indiquer la progression des appels. Normalement, les réactions sont bout à bout (à l'exception de 100 Trying). L'objectif principal des réponses d'information est d'arrêter la retransmission de INVITER demandes
2XX	Cette classe de réponses sert à indiquer qu'une demande a été acceptée. exemple : 200=’Ok’, 202=’acceptée’
3XX	En général, cette classe de réponses sont envoyés par les serveurs de redirection en réponse un INVITE. Ils sont aussi appelés classe de réponses redirigées. Redirection, une action doit avoir lieu afin de valider la requête
4XX	Erreurs du client, la requête contient une syntaxe erronée ou bien elle peut être traitée par ce serveur. Les réponses d'erreur client indiquent que la demande ne peut être satisfaite et que des erreurs sont identifiées du côté UAC. Les codes de réponse sont généralement envoyés par la UAS. Lors de la réception d'un message d'erreur, le client doit renvoyer la

	demande en la modifiant en fonction de la réponse.
5XX	Erreur du serveur, le serveur n'a pas réussi à traiter une requête. Cette classe de réponse est utilisée pour indiquer que la demande ne peut pas être traitée en raison d'une erreur avec le serveur. Le serveur n'a pas réussi à satisfaire une demande apparemment valide. La réponse peut contenir un champ d'en-tête Retry-After. La demande peut être essayée à d'autres endroits, car il n'y a pas d'erreurs indiquées dans la demande.
6XX	Echec général, la requête ne peut être traitée Cette classe de réponse indique que le serveur sait que la demande échouera où qu'elle soit tentée. En conséquence, la demande ne devrait pas être envoyé à d'autres endroits. Seul un serveur ayant des connaissances définitives de l'utilisateur identifié par le Request-URI devrait envoyer tous les cas possibles une réponse globale de la classe d'erreur. Dans le cas contraire, une réponse de classe d'erreur client doit être envoyé. Un champ d'en-tête Retry-After peut être utilisé pour indiquer que la demande pourrait être couronnée de succès.

2.2.4.3 Architecture SIP

L'architecture du protocole SIP s'articule principalement autours de cinq entités : terminal utilisateur, serveur d'enregistrement, serveur de localisation, serveur de redirection, serveur proxy. Cette architecture est entièrement logicielle.

❖ **Terminal**

C'est l'élément dont dispose l'utilisateur pour appeler ou être appelé. Celui-ci peut être un téléphone physique ou un téléphone logiciel encore appelé « softphone ». Le terminal a deux composants :

- L'UAS (User Agent Server) : il représente l'agent de la partie appelée. C'est une application de type serveur qui contacte l'utilisateur lorsqu'une requête SIP est reçue, puis renvoie une réponse au nom de l'utilisateur ;
- L'UAC (User Agent Client) : il représente l'agent de la partie appelante. Le client initie les appels et le serveur répond aux appels initiés par ce dernier.

❖ **Serveur proxy**

C'est le serveur qui se charge de la localisation de l'utilisateur appelé. Il initie, maintient et termine une session vers un correspondant. Il agit à la fois comme un client et comme un serveur. Au besoin il interprète et modifie les messages qu'il reçoit avant de les transmettre.

❖ **Serveur de redirection**

Il agit comme un intermédiaire entre le terminal du client et le serveur de localisation. L'UAC le sollicite pour contacter le serveur de localisation afin de déterminer la position courante d'un utilisateur.

❖ **Serveur de localisation**

Il permet de localiser l'utilisateur et contient la base de données de l'ensemble des utilisateurs qu'il gère. Cette base est mise à jour par le serveur d'enregistrement.

❖ **Serveur d'enregistrement**

C'est un serveur qui enregistre les terminaux SIP lors de l'envoie de la méthode REGISTER. Il offre la possibilité de localiser un correspond tout en gérant sa mobilité.

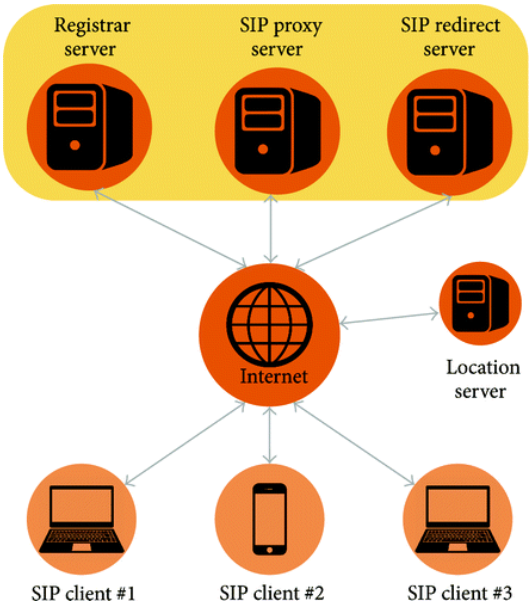

Figure 2.11: Architecture SIP

2.2.4.4 Apport de SIP dans la mutualisation des ressources

Le protocole SIP constitue un atout du fait de sa capacité à s'intégrer à d'autres protocoles standards du monde IP. En tant que standard ouvert, il offre un service modulaire, prévu pour fonctionner avec différentes applications telles que la téléphonie, la messagerie instantanée, la visioconférence [132].

En effet, Samuel OUYA dans ses travaux de thèse [132] a montré l'intérêt du protocole SIP dans un réseau IMS. À ce propos, ce protocole pourrait être envisagé dans la perspective de l'intégration des environnements d'e-santé, e-agriculture ou même d'apprentissage dans un réseau IMS.

La question de la communication de sécurité est également couverte par SIP car les mécanismes SSL (Secure Sockets Layer) / Transport Layer Security (TLS) sont déjà préparés pour une communication sécurisée entre les clients SIP. Pour les besoins de la communication M2M (domotique), SIP MESSAGE (pour les actions PUT et GET) et SIP SUBSCRIBE (pour les événements basés sur le statut, comme la notification d'alarme) peuvent être utilisés [133].

2.2.5 Présentation de WebRTC

WebRTC (Web Real Time Communication) est une technologie s'appuyant sur HTML5 et impulsée par les entreprises du web, communément appelées Over-The-Tops (OTT) [134]. La connaissance des principes de base de programmation en JavaScript et en HTML suffit largement pour intégrer des services de communication temps réel dans une page Web. Ce qui

permettrait de vulgariser l'intégration des outils de travail collaboratif et de tutorat en ligne dans les plateformes d'enseignement à distance.

2.2.5.1 Fonctionnement de WebRTC

WebRTC utilise les mêmes ports que ceux utilisés par les protocoles HTTP et HTTPS (80 et 443) ainsi que le même format d'URL par exemple ws://ucad.sn ou wss://ucad.sn. WebRTC s'appuie sur WebSocket pour établir une connexion, puis une négociation est réalisée entre le client et le serveur. Il utilise le protocole HTTP pour la phase d'initialisation de la communication. Ceci dans l'objectif d'assurer une compatibilité avec les infrastructures existantes notamment les proxys et les pare-feu [132].

L'architecture de WebRTC met en œuvre des mécanismes de communication en triangle qui impliquent un serveur Web et deux pairs. En effet, les deux navigateurs téléchargent depuis le serveur une API (Application Programming Interface) JavaScript. Le serveur Web sert uniquement à établir la connexion entre les deux navigateurs à travers le téléchargement par ces derniers de l'API WebRTC. Le flux de signalisation entre les deux navigateurs peut traverser plusieurs serveurs qui peuvent au besoin modifier, translater ou gérer ces signaux [135].

2.2.5.2 Les API de WebRTC

Les API WebRTC ont été conçues autour de trois concepts : PeerConnection, MediaStreams, DataChannel.

- **PeerConnexion**

La figure 2.10 montre l'établissement d'une connexion entre deux navigateurs WebRTC.

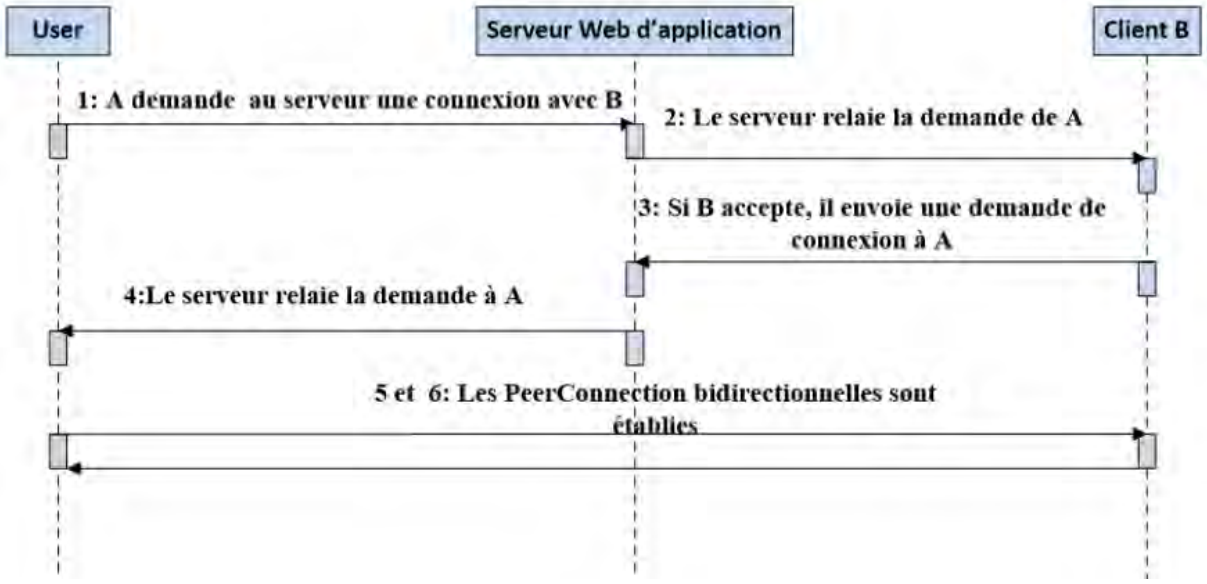

Figure 2.12: Processus de l'établissement d'une connexion

Pour établir une connexion utilisant le standard WebRTC, les navigateurs A et B doivent être connectés simultanément à la page du service et télécharger la page HTML ainsi que le code JavaScript permettant de maintenir la connexion ouverte.

Lorsque le navigateur A souhaite établir la connexion avec B, l'API instancie un objet PeerConnection qui, une fois créé, permet d'établir des flux de médias ou de données [135]. Pour une visioconférence par exemple, il est nécessaire que les utilisateurs A et B acceptent de partager leur webcam et leur microphone.

Une fois qu'un objet PeerConnection est créé par A, le navigateur envoie au serveur un paquet contenant les informations sur les médias partagés ainsi qu'une empreinte liant la connexion à A. Le serveur décode le paquet et identifie la destination de la communication. S'il s'agit effectivement d'une communication à destination de B, alors le serveur transmet la requête vers le navigateur B. Si le navigateur B accepte cette requête en provenance de A, alors le même processus a lieu entre B et A dans le but d'établir la connexion bidirectionnelle. Une fois celle-ci établie, les flux de médias ou de données peuvent être ajoutés à la connexion.

- **MediaStreams**

L'API MediaStream est une représentation abstraite des flux audio et vidéo capturés par la webcam et le microphone [136]. Pour cela, un MediaStream local doit demander l'accès aux ressources multimédia de l'utilisateur via la méthode getUserMedia. Ensuite, l'application spécifie le type de média (audio ou vidéo) auquel elle souhaite accéder et le navigateur autorise ou refuse l'accès à la ressource demandée. L'objectif à terme est d'associer un MediaStream à n'importe qu'elle source de données de transmission en continu, et pas seulement une caméra ou un microphone. La diffusion peut se faire à partir d'un disque, ou à partir des capteurs ou d'autres entrées.

- **DataChannel**

L'API DataChannel offre un moyen d'échange de données génériques bidirectionnel et pair à pair [136]. Cette composante de WebRTC permet l'échange de données telles que des images ou du texte. Des canaux de données sont créés entre pairs en utilisant un objet de type PeerConnection. Les données autres que les flux médias sont échangées via le protocole SCTP. L'intérêt de cette approche est le partage d'un même numéro de port pour les échanges des flux de données et les flux de médias.

En effet, SCTP supporte nativement plusieurs flux de données de façon bidirectionnel (jusqu'à 65536 dans chaque direction) au sein d'une association SCTP et gère les priorités. Ce qui favorise les messages de haute priorité face aux gros objets à la priorité basse.

2.2.5.3 Apports de WebRTC

Le protocole WebRTC présente plusieurs avantages par rapport aux solutions existantes du fait qu'il intègre plusieurs protocoles ouverts. Il peut par exemple utiliser le protocole SIP comme protocole de signalisation. C'est pourquoi, nous pensons que ses fonctionnalités s'adaptent à la mise en œuvre des outils de tutorat en ligne et de travail collaboratif dans une plateforme de consultation en temps réels.

Aussi, soulignons que le WebRTC peut être utilisé pour contrôler des dispositifs physiques à distance interfacés par ordinateur d'une part et, d'autre part d'observer ce que fait l'apprenant sur l'équipement si l'expérience l'exige.

En conséquence, son intégration dans les plateformes de formations à distance pourrait enrichir les scénarios d'apprentissage dans le cadre des universités virtuelles ou numériques. Pour preuve, dans [137], les auteurs montrent l'impact de son intégration dans une plateforme de formation à distance.

2.2.6 Technologies de Communications pour l'Internet des Objets

Dans la section précédente, nous avons discuté des protocoles les plus fréquemment utilisés dans le domaine IoT. Les technologies de communication IoT connectent des objets hétérogènes pour fournir des services intelligents spécifiques. En régle générale, les nœuds IoT doivent fonctionner avec une faible consommation en présence de liaisons de communication avec perte et bruit. Ces liaisons fonctionnent avec des technologies, où les messages échangés entre différents dispositifs peuvent se faire grâce à des protocoles avec des interactions en temps réel. Par conséquent, une architecture de référence unique ne peut pas être utilisée comme modèle pour tous les cas d'usage. Si un modèle de référence est identifié, il est probable que plusieurs architectectures coexisteront dans l'IoT. Tous les dispositifs IoT interconnectés utilisent différent protocoles pour l'interconnexion entre l'environnement physique et numérique, ce qui permet de collecter et de traiter les données obtenues. Dans les environnements IoT plusieurs types de protocoles de communication peuvent être utilisés. Les principaux protocoles qui permettent l'interconnexion des objets sur les réseaux IoT sont détaillés sur le tableau ci-dessous.

Tableau 2.6: Technologies de communications pour l'IoT

Protocole	Standard	Fréquences	Portée	Débit
Wifi	802.11	2.4Ghz, 5Ghz	50-125m	150Mbps-1.3 Gbps
Bluetooth LE	4.2-5.0	2.4Ghz	50-240m	1-50Mbps
LoRaWAN	LoRaWAN	variées	2-15km	0.3-50kbps
ZigBee	IEEE802.15.4	2.4Ghz	10-100m	250kbps
Z-Wave	ITU-T G.9959	900MHz	30m	9.6/40/100kbit/s
Cellular	GSM/GPRS/ 2G/3G/4G /LTE-M/ Clean Slate	900/1800/ 1900/2100MHz	35km GSM - 200km HSPA	35-170kps (GPRS), 120-384kps (EDGE), 384kps-2Mbps(UMTS), 600kbps-10Mbps (HSPA), 1-10Mbps (LTE)
NFC	ISO/IEC 1800-3	13.56MHz	10cm	100-420kbps
Sigfox	Sigfox	900MHz	10-50km	10-1000bps
Neul	Neul	900MHz (ISM), 458MHz (UK), 470-790MHz (White space)	10km	100kbps
LowPAN	RFC6282	2.4Ghz, ZigBee, low-power (sub-1GHz)	30-100m	20/40/250kbps.
DASH7	ISO 18000-7	433 MHz	1000m	200kbps
LTE-M			15km	15kbps-1Mbps
Clean Slate IoT			15km	15kbps-1Mbps

2.2.7 Une évaluation de performance de WebRTC sur LTE

L'évolution des infrastructures de télécommunications et des équipements mobiles avec l'arrivée du très haut débit a engendré de nouveaux services, en particulier audiovisuels. Les usagers utilisent de plus en plus les services multimédias tels que le streaming vidéo, la visioconférence, le télétravail...avec une mobilité garantie [138]. Une question reste cependant importante pour utiliser ces services de manière optimale : la qualité de service. Les chercheurs se sont donnés pour objectifs de tester le comportement du réseau mobile qui offre le plus le haut débit aujourd'hui (4G) pendant un appel audiovisuel entre deux terminaux à travers le WebRTC.

La LTE-A est la quatrième ère de la génération mobile qui succède aux normes 2G et 3G. Elle offre des débits environ 5 fois supérieurs en réception et près de 10 fois supérieurs en émission par rapport à la 3G+. Il s'agit d'un simulateur de réseau à événements discrets pour les systèmes Internet publié sous licence GNU GPLv2. NS-3 permet des expériences de simulation en fournissant des modèles réalistes sur le fonctionnement et l'exécution des réseaux de données par paquets. NS-3 fournit également la prise en charge de plusieurs modèles et protocoles comme Wi-Fi, WiMAX et LTE, pour n'en citer que quelques-uns. En particulier, le module LTE qui a été utilisé pour ce banc d'essai est conçu pour supporter l'évaluation de plusieurs aspects des éléments LTE, à savoir la gestion des ressources radio, l'ordonnancement des paquets QoS, la coordination entre les interférences cellulaires, etc. Il comprend

parfaitement la pile de protocoles de radio LTE (PDCP, RLC, MAC, PHY), résidant entièrement dans l'équipement utilisateur (UE) et l'e-NodeB (eNB).

L'analyse des performances a été réalisée en tenant compte du débit de l'utilisateur, du taux de perte de paquets et de la gigue. Pour analyser les appels de flux dans des conditions différentes, quatre scénarios ont été envisagés. Un premier idéal servant de point de référence où il n'y a ni interférence, ni processus de mise en file d'attente de paquets. Pour les trois autres scénarios, des caractéristiques réalistes sont progressivement ajoutées (des files d'attente de paquets, des phénomènes d'atténuation, etc.). Le dispositif expérimental virtuel comprend deux équipements utilisateurs équipés d'un navigateur ; un serveur WebRTC ; un réseau d'accès LTE composé de 2 e-NodeB et un cœur de réseau EPC.

Pour chacun des 4 scénarios, 30 appels sont effectués avec une durée moyenne de 300 secondes. Les deux équipements utilisateurs sont rattachés à deux e-NodeB situés à 10km l'un de l'autre. A l'instant t=0 (début de l'expérience), les UE se trouvent à 700 mètres chacun de leur e-NodeB respectif. Ensuite, les UE se déplacent suivant un chemin aléatoire personnalisé généré dans Matlab. Les codecs audio et vidéos utilisés pendant l'expérience sont VP8 et Opus.

Les résultats expérimentaux ont montré la détérioration de la qualité des flux multimédias lorsque des configurations plus réalistes sont envisagées. Dans le premier scénario témoin, les appels audio/vidéo WebRTC entre deux utilisateurs mobiles n'ont quasiment connu aucune perte de paquets et la communication était très fluide. En effet, plus de 50% du flux WebRTC atteint un débit de 170 Kbit / s dans le scénario n°1. En considérant des configurations plus réalistes, le pourcentage du flux WebRTC affichant des valeurs de débit plus élevées diminue.

Bien que ce travail montre que les appels audio/vidéo souffriront d'une dégradation de la qualité en fonction des paramètres cités plus haut, il est indéniable que de nos jours, LTE et WebRTC représentent deux éléments clés dans le domaine des services multimédias mobiles. Tous les smartphones étant aujourd'hui équipés de navigateurs d'une part, et d'autre part, sachant que le réseau LTE utilise l'IMS pour le traitement de la signalisation, n'est-il pas envisageable de mettre en place un cœur de réseau IMS afin d'y tester des services comme la télésurveillance des patients ou la télémédecine ?

Dans [139], Luis López-Fernández et les autres chercheurs présentent Kurento, un serveur multimédia WebRTC open source et ses APIs clients destinées à simplifier le développement d'applications pour les plateformes web et les smartphones avec des

fonctionnalités dites « rich media ». Au regard de ses fonctionnalités et en comparaison avec 4 autres serveurs multimédias de référence dans le domaine scientifique (Jitsi, Janus, Medooze, Licode), Kurento peut être un outil puissant pour les développeurs Web.

Il convient de rappeler que le WebRTC est cette technologie qui a pour ambition d'apporter la communication en temps réel (RTC) sur le Web. Aujourd'hui, la technologie est supportée par la plupart des navigateurs : chrome, firefox, Opera. Même Safari et Edge de Microsoft qui étaient réticents sont en train d'intégrer cette fonctionnalité dans leurs nouvelles versions. A l'origine, les applications WebRTC étaient basées sur l'architecture poste-a-poste (peer-to-peer). Cette architecture n'est plus suffisante pour offrir des services avancés tels que les communications de groupe, l'enregistrement de flux média, la réalité augmentée, la vision assistée par l'ordinateur, etc. Il devient donc nécessaire d'ajouter un intermédiaire : c'est le serveur multimédia. Les chercheurs soulignent qu'il n'y a pas encore une définition normalisée du serveur multimédia, mais admettent la définition suivante dans le cadre de leur travail : « un serveur multimédia est juste la partie serveur d'un système multimédia avec une architecture de type client/serveur. » Le serveur multimédia apporte donc des fonctionnalités avancées de diffusion et de traitement audio/visuel aux applications en jouant un rôle intermédiaire. Sur la base de cette définition, il est possible d'affirmer que les technologies des serveurs multimédias ont émergé dans les années 1990, catalysées par la vulgarisation des services de vidéos numériques. La plupart des serveurs multimédias étaient conçus pour fournir la distribution et le transport de contenus multimédias de deux grandes façons : les serveurs multimédias de diffusion et les serveurs multimédias de communication en temps réel.

La première catégorie concerne surtout les applications telles que la vidéo à la demande n'exigeant pas du temps réel. Les protocoles de transport utilisés diffèrent donc de la deuxième catégorie compte tenu des contraintes sur la transmission des paquets.

La deuxième catégorie est propre aux communications bidirectionnelles en temps réel. Les protocoles de transports utilisés ici doivent prendre en compte un temps de latence très faible pour garantir le temps réel. Les chercheurs soulignent que sur le marché aujourd'hui, tout serveur multimédia de communication en temps réel fournit au moins l'une de ces fonctionnalités :

- Communication de groupe audio/vidéo : permettre à un groupe d'utilisateurs de communiquer de manière synchrone.

- Archivage de médias : capacité du serveur à enregistrer des flux multimédias dans des dépôts ou référentiels. Ces flux pourront être récupérés plus tard pour la visualisation.
- Interopérabilité : il s'agit de fournir une interopérabilité des médias entre différents réseaux ayant des formats de média ou des protocoles incompatibles. A titre d'exemple, l'interconnexion entre les navigateurs WebRTC et les systèmes VoIP existants faite par des passerelles WebRTC.

Dans le développement de logiciels, un module fait référence à un bloc fonctionnel spécifique caché derrière une interface. La modularité est définie comme le caractère de ce qui est modulaire, c'est-à-dire formé d'un ensemble de modules. Un serveur multimédia modulaire doit donc avoir un ensemble de modules avec une forte cohésion et une possibilité de couplage entre chacun des modules.

Selon ces auteurs [140], il ne peut y avoir de modularité totale sans les 4 propriétés suivantes :

- **Isolable :** lorsque les modules fonctionnent ensemble dans un système, l'état interne d'un module ne doit pas affecter directement l'état des autres ;
- **Abstraction :** l'état interne d'un module doit être caché pour le concepteur du système et pour les autres modules. Dans le logiciel, l'abstraction est généralement réalisée à travers des interfaces qui limitent la façon dont le module interagit avec le monde externe grâce à un ensemble limité de primitives ;
- **Composable :** ces interfaces abstraites devraient permettre aux composants de se recombiner dans diverses combinaisons pour satisfaire les besoins spécifiques des utilisateurs. En d'autres termes, les modules devraient se comporter comme des blocs de construction, et le rôle du concepteur du système est de créer la topologie d'interconnexion appropriée parmi eux pour fournir la logique souhaitée ;
- **Réutilisable :** si un module fournit une capacité spécifique, tout système nécessitant cette capacité peut réutiliser le module sans qu'il soit nécessaire de réintroduire à nouveau la capacité ;
- **Extensible :** la plate-forme devrait fournir des mécanismes intégrés pour créer et brancher des modules supplémentaires, en étendant les capacités déjà présentes.

Les auteurs de [139] signalent à travers le tableau comparatif suivant, que Kurento est le seul serveur multimédia intégrant l'ensemble de ces fonctionnalités comme l'indique ce tableau comparatif issu de leur travail.

Tableau 2.7: Comparaison de la modularité au sein de plusieurs serveurs médias

	Isolation	Abstraction	Composabilité	Réutilisabilité	Extensibilité
Jitsi	Non	Oui	Non	Non	Non
Janus	Oui	Oui	Non	Oui(partiel)	Oui
Medooze	Non	Oui	Non	Non	Non
Licode	Oui(Partiel)	Oui	Non	Non	Non
Kurento	Oui	Oui	Oui	Oui	Oui

Pour rendre possible la communication de groupe, les serveurs multimédias utilisent dans leur majorité deux stratégies : Le mixage de médias et le transfert de média.

Le mixage de médias se fait avec la topologie MMM (Media Mixing Mixer). Dans cette dernière, un participant à une session de communication de groupe envoie un flux multimédia (son propre flux audio et/ou vidéo) et reçoit un flux multimédia (flux audio et/ou vidéo mixé de tous les participants).

Le transfert de média est fait grâce à une topologie incluant une boîte intermédiaire de transfert sélectif. On parle de SFU (Selective Forwarding Units) ou de MSM (Media Switching Mixer). Dans cette topologie, un participant envoie un flux multimédia (le sien) et reçoit N-1 flux (les flux des autres participants) avec N le nombre total de participants.

Au total, il y a donc trois topologies (MMM, SFU, MSM) utilisées pour offrir la communication de groupe au sein des serveurs multimédias. De tous les serveurs étudiés dans ce travail, seul Kurento offre les trois topologies à la fois via des API.

Pour des raisons d'interopérabilité avec d'autres systèmes de communication, il est important pour un serveur multimédia d'offrir une fonctionnalité de transcodage. Contrairement aux autres serveurs multimédias de communication en temps réel, Kurento se charge lui-même de cette fonctionnalité avec une abstraction totale pour le développeur et les utilisateurs. Pour

ce faire, un module appelé « Agnostic media » existe au sein du serveur et fait toutes les opérations de transcodage de manière transparente.

Kurento est le seul serveur multimédia WebRTC qui implémente les 5 caractéristiques fondamentales d'un serveur modulaire. Il peut être utilisé pour des applications dans le E-santé, le E-learning, le E-agricultureles jeux vidéo, etc.

Afin de garantir la modularité totale, Kurento a mis en place le concept de « Media Element ». Un « media element » est un module fournissant une fonctionnalité bien précise. Au retrouve au sein des API Kurento un ensemble de « media element » selon la classification suivante :

- endpoint : un média élément avec la capacité de communiquer avec l'extérieur. (webrtcendpoint) ;
- filter : tout média élément avec la capacité de traitement multimédia (détection faciale) ;
- call hub : un média élément qui rend possible la communication de groupe ;
- media pipeline : un graphe de médias éléments connectés.

Les modules de Kurento sont basés sur Gstreamer et sur OpenCV. Il est intéressant de passer en revue l'architecture détaillée des médias éléments de Kurento.

A l'intérieur d'un média élément se trouve un composant appelé « KMSMediaElement » qui est la classe de base de tous les médias éléments. Pour échanger avec d'autres medias elements au sein d'un media pipeline, il est important d'utiliser ce qu'on appelle le « pad ». Le KMSMediaElement s'occupe de la gestion des « pads et bins », termes propres au jargon de Gstreamer. En d'autres termes, la gestion des entrées/sorties à l'intérieur du media element. Un pad d'entrée est appelé sink et un pad de sortie est appelé source. Le bin est un objet Gstreamer qui s'occupe du traitement de médias. Dans ce cas précis, l'agnostic bin s'occupe du transcodage.

Pour exploiter les fonctionnalités de Kurento Media Server, il existe des API implémentées sous forme de clients. Un développeur peut utiliser Kurento avec les scénarios suivants :

- Utilisation directe du client Javascript dans un navigateur WebRTC compatible
- Utilisation du client Java Kurento dans un serveur d'application JEE
- Utilisation du client Kurento Javascript dans un serveur Node.js.

Pour finir, les chercheurs ont évalué l'évolutivité de Kurento Media Serveur. Ils se sont posé la question de savoir si Kurento est capable de supporter une montée en charge tout en

maintenant la même qualité de services. Les résultats sont positifs et confirment que Kurento est un serveur multimédia WebRTC exceptionnel.

2.2.8 IMS

Le but de ce paragraphe est de présenter une définition de l'IMS, les architectures de réseau et de service IMS avec les concepts sous-jacents, les entités impliquées et leurs fonctionnalités.

2.2.8.1 Définition de l'IMS

Le sous-système multimédia IP ou IMS est un réseau 3GPP (release 5) à part entière qui communique avec le cœur réseau EPC et avec les domaines paquet de l'UMTS et GSM (il fait abstraction du réseau d'accès) afin de controler les services temps réel IP multimédia tel que la voix sur IP. L'IMS est un système complexe qui nécessite des investissements importants de la part de l'opérateur. Cependant il est très approprié pour l'acheminement de la voix et SMS sur LTE et peux être considéré comme une version sophistiquée du VoIP de 3GPP.

Ses composants les plus importants sont les fonctions de commandes de session d'appel ou Call State Control Function (CSCF) dont il existe trois types :

- **S-CSCF** (Server Call State Control Function) : c'est le point d'accès dans le réseau IMS d'un UE. Il est chargé d'initier, de contrôler et de maintenir la session des terminaux.
- **P-CSCF** (Proxy Call State Control Function) : C'est le point d'entrée et de contact à IMS pour les UEs. Il transmet des messages de signalisation SIP au S-CSCF résidentiel de l'utilisateur, compresse les messages de signalisation que l'UE échange avec l'IMS de manière à réduire leur charge sur le réseau de transport de LTE et sécurise les messages en les cryptants et en protégeant leur intégrité. Il communique également avec le PCRF afin de garentir la qualité de service.
- **I-CSCF** (Interrogating Call State Control Function) : C'est le premier point de contact pour les messages de signalisation qui d'un autre IMS.

❖ **Objectifs**

Les objectifs de l'IMS sont :

- Diversifier les services à offrir à l'utilisateur ;
- Faire la convergence Fixe/Mobile ;
- Renouveler le rôle des opérateurs et fournisseurs d'accès ;

- Offrir de nouveaux services aux utilisateurs ;
- Migrer vers le tout IP ;
- Procéder à la standardisation et la simplification de l'administration des réseaux.

2.2.8.2 Architecture de l'IMS

IMS repose sur une architecture modulaire qui permet de distinguer des niveaux de traitements différents. Quatre couches peuvent être identifiées (accès, transport, contrôle, application), chacune d'elles étant liée à un domaine spécifique comme le montre la figure 2.13 ci-dessous.

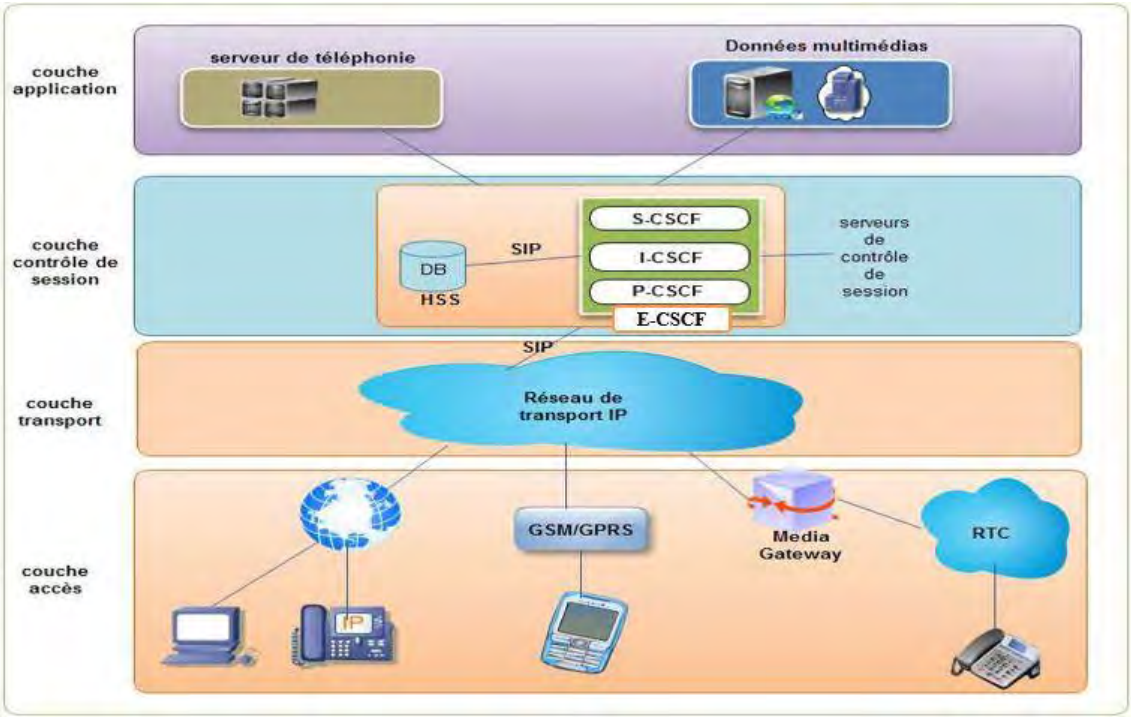

Figure 2.13: Architecture en couche de l'IMS

Chaque couche est indépendante. De ce fait, il est possible, par exemple, d'ajouter librement de nouveaux services dans la couche applicative, sans tenir compte du réseau d'accès que les utilisateurs ont employé, ni du terminal qu'ils ont utilisé.

- **La couche physique**

La couche physique est une couche d'accès de l'architecture IMS qui permet un large choix de terminaux aux utilisateurs. Elle regroupe les fonctions et les équipements permettant de gérer l'accès des utilisateurs au réseau, selon la technologie.

- **La couche transport**

La couche de transport s'occupe de l'acheminement du trafic voix ou données dans le cœur du réseau, selon le protocole utilisé. Elle permet une connectivité de bout en bout entre

les différents interlocuteurs. C'est le réseau IP qui est utilisé dans cette couche. Elle est formée d'un maillage de commutateurs et de routeurs qui assurent, dans le réseau IP, le routage des données multimédias, à l'exclusion des informations de signalisation, qui sont à la charge de la couche de contrôle. C'est ici que les terminaux se connectent au réseau IP, indépendamment du média de transmission. Les technologies d'accès Wi-fi, cable, GPRS, UMTS, etc. sont alors traitées à ce niveau. Un des principaux enjeux au niveau de cette couche est de réaliser la convergence entre réseau à routage par paquet et réseaux à routage par circuit, ou autrement dit entre téléphone commuté et réseau utilisant TCP/IP.

- **La couche contrôle de session**

Elle assure la gestion et le contrôle du réseau. Elle est en charge de tous les messages de signalisation dans le réseau, permettant d'ouvrir, de maintenir, de modifier et de terminer une session entre utilisateurs. C'est la partie intelligente du modèle, qui offre toutes les fonctionnalités de gestion des utilisateurs et constitue le véritable socle de l'IMS et, ce qui signifie que, malgré leur distinction fonctionnelle, rien n'empêche de les implémenter (toutes ou certaines) au sein d'un même équipement. La couche de contrôle est constituée de différentes entités logiques : le P-CSCF, l'I-CSCF, le S-CSCF, l'E-CSCF et le HSS.

Le Proxy CSCF (P-CSCF)

Un proxy-CSCF est un proxy SIP qui est le point de contact entre un terminal et le réseau IMS. Ses missions consistent notamment à controler l'accès et à établir une connexion sécurisée avec le terminal. Son adresse est découverte par l'utilisateur lors d'une phase de « CSCF discovery». Il agit comme intermédiaire entre l'abonné et le I-CSCF.

Certains réseaux peuvent utiliser un Session Border Controller (SBC) pour cette fonction. Le P-CSCF possède à sa base un SBC spécialisé pour l'interface utilisateur-réseau qui non seulement protège le réseau, mais aussi le terminal IMS. L'utilisateur de SBC supplémentaires entre le terminal IMS et le P-CSCF est donc inutile et n'est pas réalisable du fait que la signalisation est cryptée à partir de cette liaison. Le terminal peut découvrir son P-CSCF à l'aide d'un serveur DHCP.

L'Interrogating CSCF (I-CSCF)

L'Interrogating CSCF a comme principales fonctions de déterminer le S-CSCF auquel l'abonné peut se connecter et transmettre les messages entre le P-CSCF et le S-CSCF, un peu comme une passerelle.

Les fonctions réalisées par cette entité sont :

- La localisation du S-CSCF concerné par la session par consultation de la base HSS (load balancing de S-CSCF) ;

- La garentie de sécurité entre le Visited network et le Home network.

Le Serving CSCF (S-CSCF)

Le serving CSCF est l'équipement qui a pour rôle de finaliser l'authentification de l'utilisateur et lui fournir les services opérationnels. Il fournit des informations de routage, de facturation, maintient l'état de la session en contrôlant un timer, interroge le HSS pour vérifier les droits utilisateurs vis-à-vis d'un service, etc.

Le Home Subscriber Server (HSS)

Le HSS est la base de données des abonnés de l'IMS et des services associés (à l'instar du HLR pour les réseaux mobiles). Les comptes utilisateurs, leurs profils et droits d'accès, le nom du S-CSCF associé sont stockés dans cette base. Il communique avec le CSCF pour fournir, temporairement, une copie du profil utilisateur.

Policy Decision Function (PDF)

Dans l'objectif d'assurer la mise en œuvre de politiques de provisionning, de routage ou de QoS, la release 5 des spécifications 3GPP a prévu l'utilisation d'une plate forme de distribution de politiques conforme COPS. Dans ce contexte, la fonction PDF est une entité fonctionnelle dont le rôle est d'assurer la distribution des politiques de services locales (Service Based Local Policy : SBLP). A cet effet, la release 5 a intégré cette fonction dans le P-CSCF, tandis que pour la release 6 l'entité PDF est prise en charge par un bloc fonctionnel indépendant du P-CSCF.

Passerelles et Contrôle de passerelles

Les Gateways (passerelles) ont un rôle essentiel : elles assurent non seulement l'acheminement du trafic, mais aussi l'inter fonctionnnement avec les réseaux externes et avec les divers réseaux d'accès.

La Média Gateway (MG)

La Media Gateway est située au niveau du transport des flux entre le réseau RTC et les réseaux en mode paquet, ou entre le cœur de réseau NGN et les réseaux d'accès. Elle a pour rôle le codage et la mise en paquets du flux média reçu du RTC et vice versa (conversion du trafic TDM IP) et aussi la transmission, suivant les instructions du Media Gateway Controller, des flux media reçus de part et d'autre.

Signilling Gateway (SG)

La fonction Signalling Gateway est de convertir la signalisation échange entre le réseau NGN et le réseau externe interconnecté selon un format compréhensible par les équipements chargés de la traiter, mais sans l'interprèter (ce rôle étant dévolu au Media Gateway Controller). Notamment, elle assure l'adaptation de la signalisation par rapport au protocole de transport

utilisé. Cette fonction est souvent implémentée physiquement dans le même équipement que la Media Gateway, d'où le fait que ce dernier terme est parfois employé abusivement pour recouvrir les deux fonctions MG + SG.x

Breakout Gateway Control Function (BGCF)

La fonction de contrôle de passerelles de dérivation (BGCF) détermine le réseau de destination pour lequel le réseau PSTN doit se connecter ; elle choisit détermine le MGCF local ou la BGCF homologue et fournit la sécurité par l'autorisation des réseaux homologues.

Media Gateway Control Function (MGCF)

Le contrôle des ces passerelles est assuré par la MGCF qui assure les fonctions suivantes : Conversion du protocole ISUP provenant du RTC en protocole SIP, Contrôle les parties de l'appel qui maintient le contrôle de conexion pour les canaux media, Comminique et dialogue avec la CSCF, Sélectionne le CSCF en fonction du routing number pour les appels entrants.

- **La couche application**

La couche application est la dernière couche de l'architecture IMS. Elle regroupe les services applicatifs. Trois plateformes de services ont été standardisées pour offrir les services à valeurs ajoutées. Tout service est exécuté par un serveur applicatif, en liaison avec les équipements de la couche session par l'intermédiare des protocoles SIP et Diameter, aussurant ainsi la sécurité des utilisateurs.

Il y a trois plates-formes de services standarisées : (1) SIP application server (**SIP-AS**), (2) Open Service Access (**OSA**), Service Capability Server (**SCS**), (3) et IP Multimedia Server Switching Function (**IM-SSF**).

Les services offerts par ces plateformes sont des services à valeurs ajoutées (value-added services VAS) ou des services spécifiques à l'opérateur. L'OSA SCS et l'IM-SSF ne sont pas serveurs d'applications proprement dit. Ce sont plus passerelles d'accès aux autres environnements de services. L'OSA SCS et l'IM SSF interfacent respectivement avec l'OSA application server et le CAMEL Service Environment (CSE). Du point de vue du S-CSCF, cependant, ils présentent tous le même comportement de l'interface ISC.

2.2.8.3 Protocoles utilisés dans l'IMS

Les protocoles utilisés dans l'IMS sont :

SIP : est le protocole fédérateur de l'architecture IMS. Il est en quelque sorte la glue qui permet aux différents composants de communiquer entre eux de manière homogène.

Son choix, par rapport à tout autre protocole de signalisation, n'est pas anodin puisqu'il pérennise SIP au détriment de H.323, jugé trop lourd et trop coûteux. Il marque ainsi la confiance des industrieles associés, en particulier SDP, pour la description des sessions, et RTP/RTCP, pour le transport en temps réel des flux de données multimédias.

Diameter : décrit dans la RFC 3588 de l'IETF, est un protocole de type AAA (Authentification, Authorization, Accounting). Issu du protocole RADIUS (Remote Authentication Dial-In User Service), Diameter constitue une évolution de la technologie AAA. Il est employé pour la sécurisation des communications, notamment lors de l'enregistrement des utilisateurs et lorsque les profils utilisateurs sont transférés d'une base de données vers les serveurs de traitement. Il fonctionne en mode client / server, donc sous la forme de requêtes et de réponses.

COPS (Common Open Policy Service) : est un protocole flexible permettant la mise en place de politiques par une entité centrale appelée PDP (Policy Decision Point), qui sont appliquées sous formes de régles dans des entités appelées PEP (Policy Enforcement Point). COPS sert notamment à permettre aux opérateurs de garantir une qualité de service dans une architecture IMS.

2.3 CONCLUSION

Nous avons étudié les protocoles d'applications pour l'internet ses objets susceptibles d'être mises à contribution dans la mise en place d'un middleware IoT pour le E-santé, E-Agriculture et la formation à distance.

Dans le contexte IoT, AMQP est le plus approprié pour le plan de contrôle ou les fonctions d'analyse basées sur le serveur. Par conséquent, ce n'est pas un candidat approprié pour la transmission de données M2M.

L'étude des protocoles de signalisation révèle que le protocole XMPP est le mieux adapté à la gestion de la découverte dynamique des ressources partagées au sein d'un espace intelligent.

De plus, la nature ouverte de SIP fournit une base solide pour résoudre les problèmes spécifiques au domaine d'automatisation domestique / industriel. Le SIP constitue un protocole de communication sûr et fiable pour les services IoT distants.

Par ailleurs, nous avons mis en relief l'intérêt d'intégrer la norme WebRTC dans un middleware IoT pour faciliter le partage des données recueillis par les capteurs et la communication entre acteurs.

3 Middleware IoT et mecanisme de gestion de trafic

3.1 Introduction

Ce chapitre traite les middlewares dans le domaine de l'internet des objets. Nous définirons les concepts et préliminaires des middlewares IoT afin de situer le contexte de l'usage de certains d'entre eux dans cette thèse.

Nous présentons les principaux modèles middlewares IoT, puis identifions lequel d'entre eux est le plus utilisé dans le domaine des e-santé, e-formation et e-agriculture. La connaissance des modèles middlewares nous permettra de cerner les exigences technologiques inhérentes à la conception d'une stratégie de mutualisation.

Notre approche de gestion de la QoS se traduit par la mise en œuvre de mécanismes ayant un impact sur les performances du système considéré. Ces mécanismes sont activés dans le cadre d'une architecture de gestion conduisant les entités de niveau Middleware à être adaptées de façon dynamique pour qu'elles répondent au mieux aux contraintes de QoS des applications et de l'opérateur du système.

3.2 Concepts et préliminaires de middleware IoT

Généralement, le middleware joue le rôle d'un interprète qui comble le fossé entre les exigences de haut niveau des différentes applications IoT et la complexité des différentes opérations dans le matériel du nœud de capteur sous-jacent [140]. La conception d'un middleware IoT doit relever plusieurs défis [141] [142] :

- **Gestion des ressources et de la batterie limitée** : les middlewares conçus pour les objets intelligents devraient fournir un mécanisme approprié pour utiliser une mémoire limitée et assurer une consommation d'énergie efficace ;
- **Évolutivité**, mobilité et topologie de réseau dynamique : les intergiciels doivent maintenir les performances requises pendant la croissance de l'environnement réseau.
- **Hétérogénéité :** l'hétérogénéité entre le matériel, les dispositifs de communication et les opérations de configuration doit être gérée par le middleware ;
- **Intégration dans le monde réel** : le middleware doit fournir des services en temps réel qui doivent être adaptés aux éventuels changements et à la mise à jour des données ;

- **Agrégation de données** : L'agrégation de données au sein du réseau garantit que les données redondantes ne sont pas générées, ce qui réduit les coûts liés à l'utilisation de la mémoire et l'énergie nécessaire au traitement ;
- **Connaissance de l'application** : le middleware doit inclure des mécanismes permettant d'injecter la connaissance de l'application de l'infrastructure IoT ;
- **Qualité de service (QoS) :** est importante tant au niveau de l'application que du réseau, car elle définit la précision des données, la couverture et la tolérance ;
- **Sécurité :** il est nécessaire d'assurer la confidentialité et l'intégrité des données, car les informations collectées transmises sur le réseau peuvent facilement être piratées par des intrusions malveillantes ;
- **Tolérance aux pannes :** l'intégration de méthodes de récupération dans le système est nécessaire pour permettre des réseaux résistant aux pannes.

Les approches middleware peuvent être classées en deux groupes principaux, en fonction de leur fonctionnement : celles qui ne fonctionnent que dans le réseau (approches classiques) et celles qui intègrent et traitent des données qui fonctionnent uniquement en dehors du réseau [141].

3.3 Architectures middleware pour les systèmes IoT

Les solutions de middleware fournissent une infrastructure technique assurant la médiation entre deux systèmes ou plus [143]. Leur rôle historique est d'assurer le transport d'un message d'un sous-système à un autre avec un niveau de couplage plus ou moins important. Les chercheurs considèrent que les middlewares sont une solution appropriée pour combler le manque d'hétérogénéité et de facilité de gestion des capteurs, car ils fournissent une couche abstraite positionnée entre la couche réseau et la couche d'application, comme illustré à la Figure 3.1. En outre, il vise à masquer les détails technologiques des technologies de communication pour permettre aux développeurs d'applications de se concentrer sur le développement des applications IoT.

3.3.1 Défis du middleware IoT

Plusieurs architectures de middleware ont été proposées dans la littérature pour tenter de relever les défis d'infrastructure et d'applications de l'IoT. Ils sont considérés comme des problèmes essentiels à résoudre avant de présenter une approche comme solution finale.

Les problèmes d'infrastructure concernent (1) le problème de l' **interopérabilité** , dans la mesure où des dispositifs hétérogènes communiquent et échangent ensemble des informations; (2) le problème de **l' évolutivité** puisqu'un grand nombre de périphériques devrait être pris en charge par l'application IoT; (3) les **interactions spontanées** d'objets et de dispositifs; (4) la **diversité des infrastructures** puisque les périphériques IoT disposent d'une infrastructure spécifique (mobile, connectée sans fil, etc.) et de ressources limitées, ce qui confère au réseau IoT un comportement dynamique ; et (5) le besoin **d' abstraction** à plusieurs niveaux, tels que la couche physique, les interfaces, le flux de données et le processus de développement.

Les défis de l'application consistent à garantir (1) **la disponibilité** des services et des informations à tout moment ; (2) un haut niveau de **fiabilité** du système qui devrait rester opérationnel même en présence de pannes ; (3) une fourniture d'informations en **temps réel**, en particulier pour des domaines critiques tels que les soins de santé ; (4) et enfin la **sécurité** et la **confidentialité**, étant donné que beaucoup d'informations sont partagées entre les composants IoT. Ces informations peuvent être privées et même personnelles, telles que des informations sur la vie quotidienne.

L'objectif commun de toutes ces initiatives de développement de middleware est de développer un cadre qui permette une couche d'adaptation en mode plug-n-play. Malgré cela, chaque architecture de middleware se concentre sur certains des défis cités précédemment et prend en compte les exigences de certaines applications qui diffèrent les unes des autres en ce qui concerne les domaines IoT qu'elles ciblent [144]. À la lumière des défis présentés, nous fournissons un aperçu de ces approches en soulignant leurs caractéristiques.

3.3.2 Présentation des approches middleware pour l'IoT

Plusieurs études sur les approches middleware de l'IoT ont été proposées, à l'instar des classifications présentées dans [145] [146]. Une étude spécifique est présentée dans [144]. Dans cette étude, les auteurs ont analysé en profondeur plusieurs solutions existantes et présenté un aperçu récent des différents types d'architecture de middleware. Grâce à cette étude, nous avons identifié les différents types de middleware (voir Figure 3.1) pour les environnements intelligents classés comme suit : applications, middleware orienté service, base de données, basé sur agent, middleware orienté message etc.

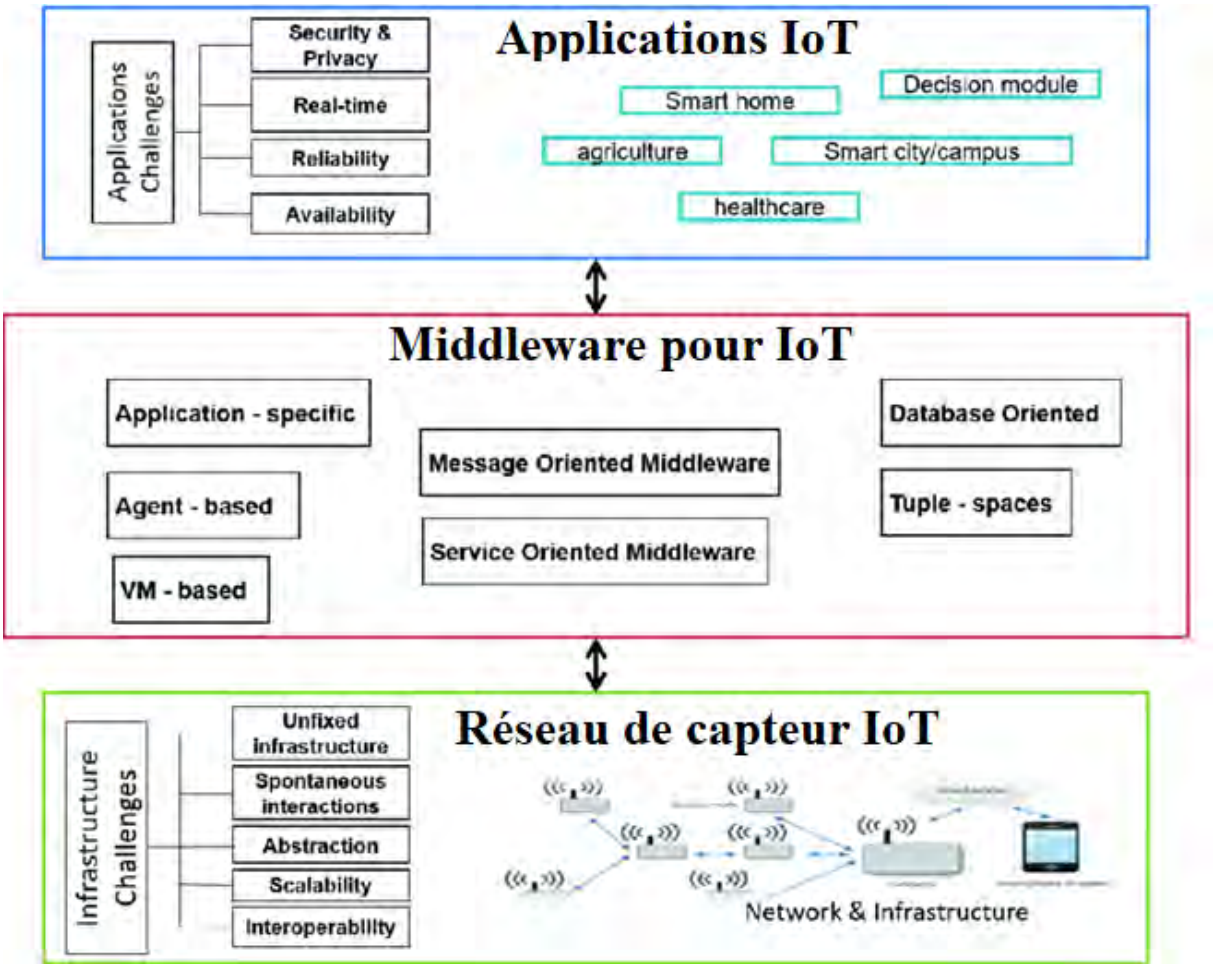

Figure 3.1: La relation entre les sources de données IoT et les applications IoT via le médiateur middleware

3.3.2.1 Middleware orienté application-spécifique

Cette approche est basée sur les besoins d'une application spécifique et se concentre sur la gestion des ressources. Ainsi, un couplage étroit existe entre les fournisseurs d'applications et de données, ce qui conduit à un middleware spécialisé. MidFusion [147] représente un exemple de cette approche de middleware. Il découvre et sélectionne le meilleur ensemble de capteurs ou d'agents de capteur pour le compte d'applications. Il fournit un algorithme de sélection de capteurs permettant de sélectionner le meilleur ensemble de capteurs en utilisant les principes des théories bayésienne et décisionnelle. Sa limitation est qu'il fournit une prise en charge de la qualité de service (QoS) uniquement pour les réseaux basés sur des algorithmes bayésiens. Nous trouverons d'autres middlewares de la même catégorie dans [148], qui porte sur les applications dans les maisons intelligentes. Il utilise des fonctions spécifiques à l'application. Les auteurs [149], proposent MiLAN, un middleware basé sur un couplage étroit entre les composants. Il permet aux applications de spécifier une stratégie de gestion du réseau et des capteurs. Ils ont

fait valoir que les besoins de l'application devraient être intégrés à la gestion du réseau dans un seul système de middleware unifié.

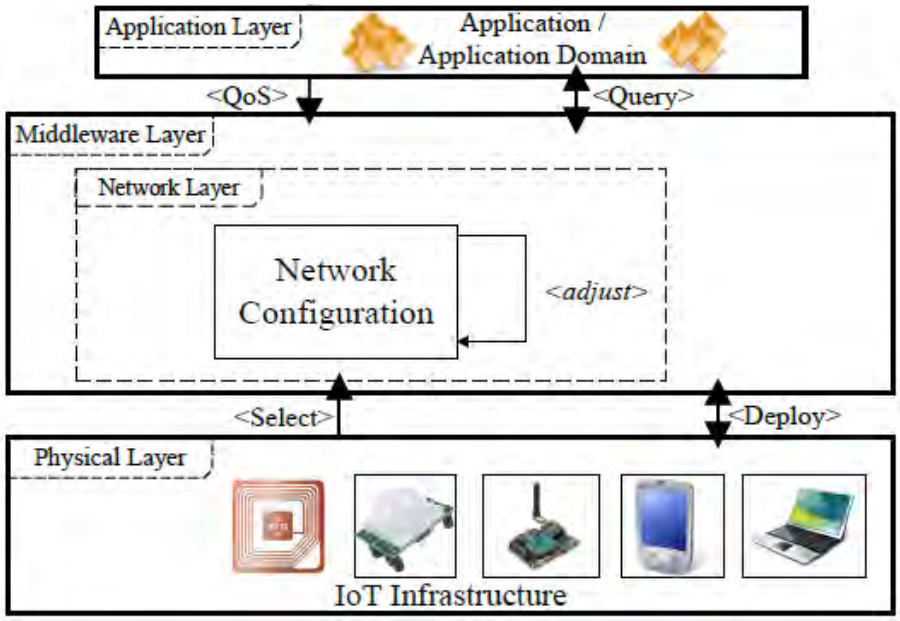

Figure 3.2:Modèle de conception général pour le middleware spécifique à l'application

Ce type d'architecture ne satisfait pas toutes les exigences des middlewares IoT en raison de son caractère de couplage étroit entre applications et fournisseurs de données. Il ne traite pas de la caractéristique d'hétérogénéité de l'environnement IoT.

3.3.2.2 Middleware basé sur l'agent

L'approche par agent ou par module [150] consiste en une division des applications en programmes modulaires pour promouvoir la distribution et l'injection sur le réseau à l'aide d'agents mobiles. Impala, Smarty messages et Agilla sont des exemples de cette approche [151] [152] [153]. Ils peuvent être mis en évidence en fournissant des systèmes décentralisés capables de répondre aux exigences de disponibilité, de fiabilité et de gestion des ressources du middleware. Cette approche peut réduire la complexité d'une architecture middleware. Cependant, il présente certaines limitations liées à son incapacité à effectuer des tâches de gestion de code et à l'imprévisibilité des agents dans le système au moment de l'exécution.

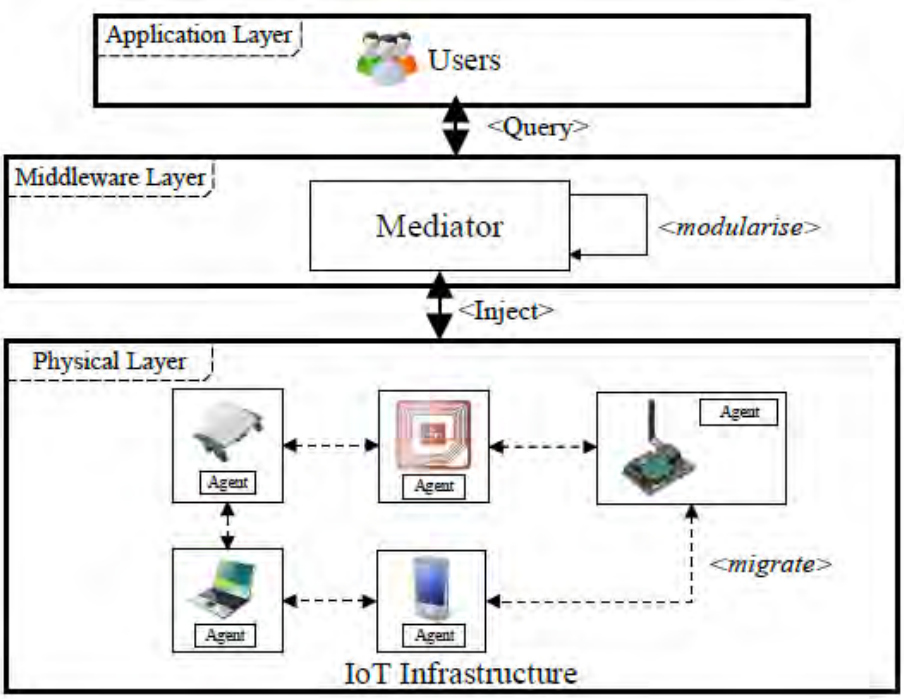

Figure 3.3: Modèle de conception générale pour les middlewares basés sur des agents

3.3.2.3 Middleware basé sur les machines virtuelles

Cette approche est flexible et contient des ordinateurs virtuels, des interpréteurs et des agents mobiles. Le middleware est alors composé de deux couches. Chaque périphérique physique est déployé en tant que machine virtuelle dans la première couche du middleware. Dans la deuxième couche, une machine virtuelle générale interprète les modules et fournit des données à l'application qui en exprime les besoins à l'aide d'une requête. Cette approche répond aux exigences architecturales telles que les abstractions de programmation de haut niveau, l'autogestion et l'adaptivité, tout en prenant en charge la transparence dans les infrastructures distribuées hétérogènes d'IoT [154] [155]. Cependant, cette approche souffre de la surcharge que les instructions échangées introduisent.

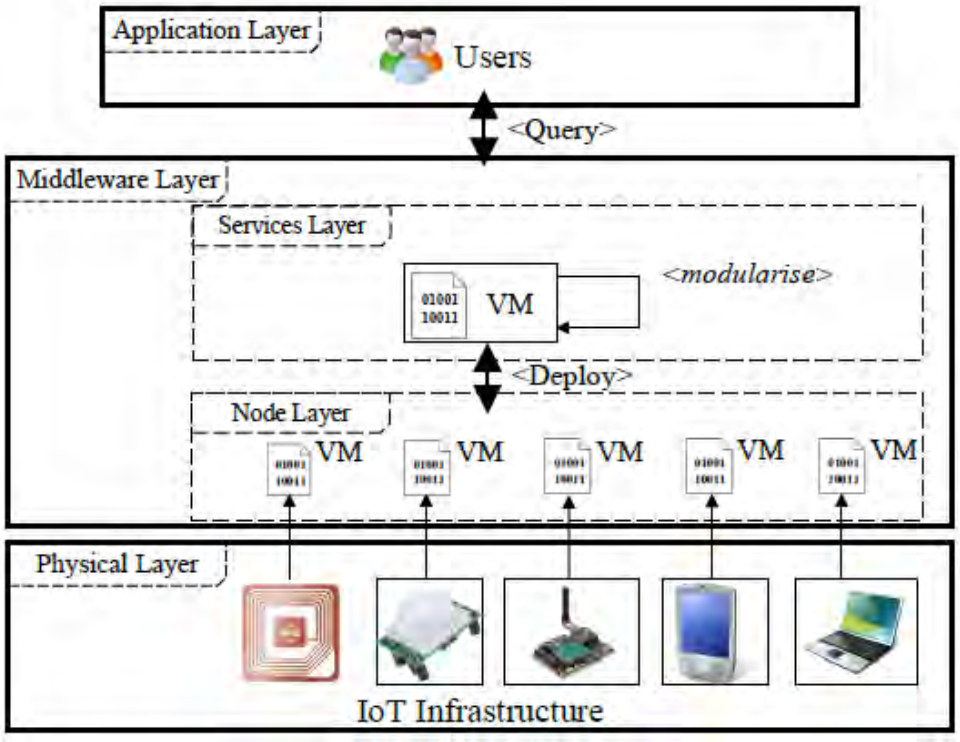

Figure 3.4: Modèle de conception général pour le middleware basé sur la VM

3.3.2.4 Middleware oriente tuple-spaces

Un tuple-space est un référentiel de données accessible simultanément [156]. Chaque périphérique de la couche physique est représenté sous la forme d'un tuple-espace dans le middleware. Tous les espaces de tuple forment un espace fédéré sur la passerelle. Cette approche convient aux appareils mobiles dans une infrastructure IoT car ils peuvent partager des données de manière transitoire dans le cadre de contraintes de connectivité de passerelle. TinyLime et TeenyLIME sont des solutions middleware tuple-space pour les réseaux ad hoc mobiles et les réseaux de capteurs [157] [158]. Bien que leur architecture flexible permette l'utilisation de middleware dans différents environnements, ils permettent de résoudre les problèmes de déconnexion fréquente et de problèmes de communication asynchrone. Cependant, ils offrent des limites en matière de gestion des ressources, d'évolutivité, de sécurité et de confidentialité.

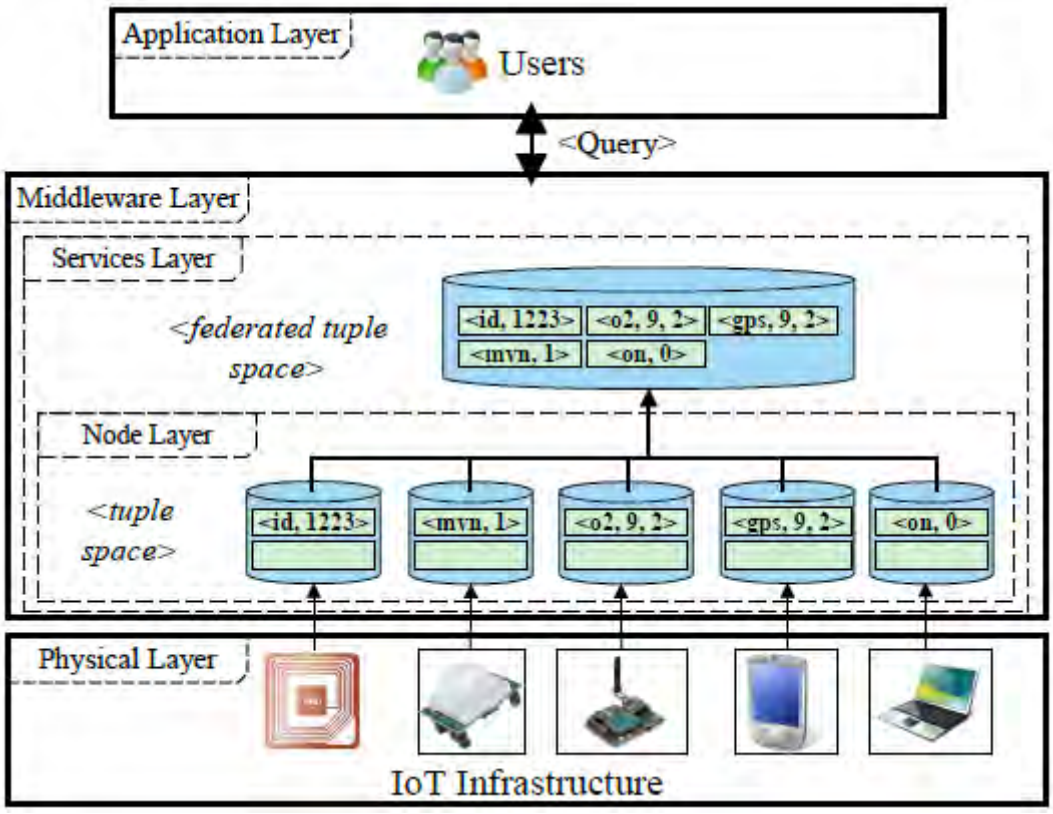

Figure 3.5: Modèle de conception général pour le middleware basé sur l'espace de triples

3.3.2.5 Middleware oriente base de données

Cette approche middleware considère l'ensemble du réseau de capteurs comme une base de données distribuée et virtuelle. Il utilise SQL comme des requêtes pour collecter des données sur le réseau. GSN est un middleware orienté base de données qui a été intégré à d'autres projets tels que OpenIoT [159] [160]. Bien que cette approche offre une bonne abstraction de programmation et un support approprié pour la gestion des données, les exigences restantes de l'IoT ne sont pas nécessairement traitées comme l'évolutivité, les interactions en temps réel et spontanées. De plus, sa nature centralisée rend difficile la gestion des caractéristiques dynamiques et d'hétérogénéité du réseau IoT.

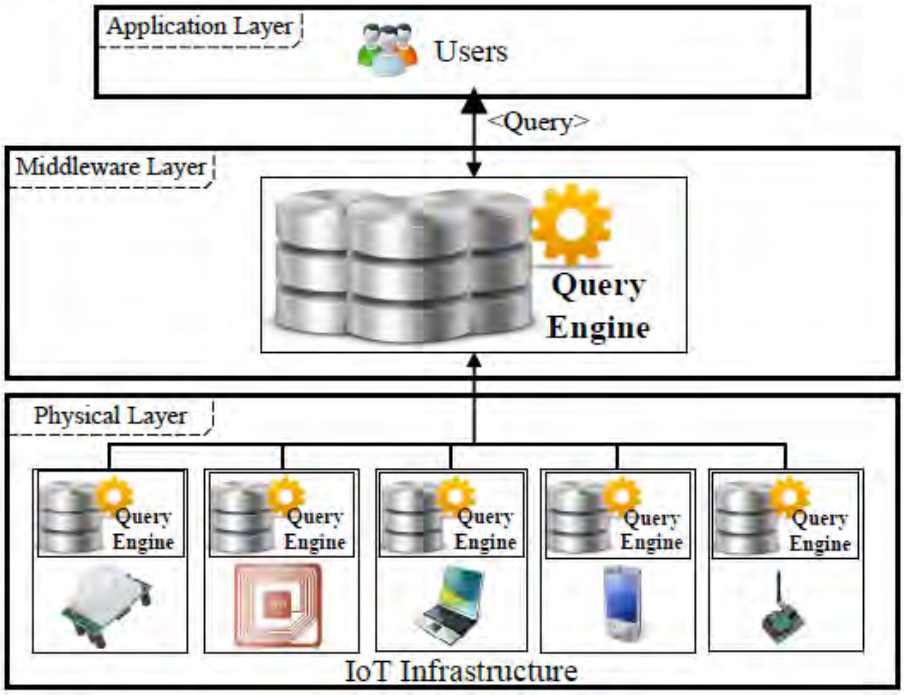

Figure 3.6: Modèle de conception générale pour le middleware orienté base middleware orienté base de données

Les architectures mentionnées ci-dessus ont été utilisées dans de nombreux projets et domaines de recherche IoT spécifiques. Cependant, l'utilisation importante des solutions d'architecture orientée services (SOA) et de middleware orienté message (MoM) est remarquable dans les projets IoT de la dernière décennie.

3.3.2.6 Middleware orientée services (SOA)

Deux tendances majeures dans le monde de l'IoT ont été observées ces dernières années [163]. Premièrement, le matériel devient plus petit, moins cher et plus puissant. Deuxièmement, l'industrie du logiciel s'oriente vers les technologies d'intégration orientées services. L'architecture orientée services (SOA) est une façon de penser et de concevoir le système d'information. Elle est traditionnellement utilisée dans les systèmes d'information d'entreprise. Le concept clé de la SOA est le service qui est un logiciel distribué invocable qui peut être rapidement modifié ou redéployé dans de nouveaux contextes, permettant aux applications de répondre rapidement aux besoins changeants des consommateurs. Dans les approches IoT basées sur la SOA, les capteurs intelligents sont décrits comme des services destinés aux applications grand public. Le point clé de ces services est leur nature encapsulée (l'interface de service est indépendante de la mise en œuvre). Les fournisseurs de services décrivent leurs services (caractéristiques des capteurs) et les exposent aux consommateurs. Le langage de description de services Web (WSDL) est la norme utilisée pour une telle description [161].

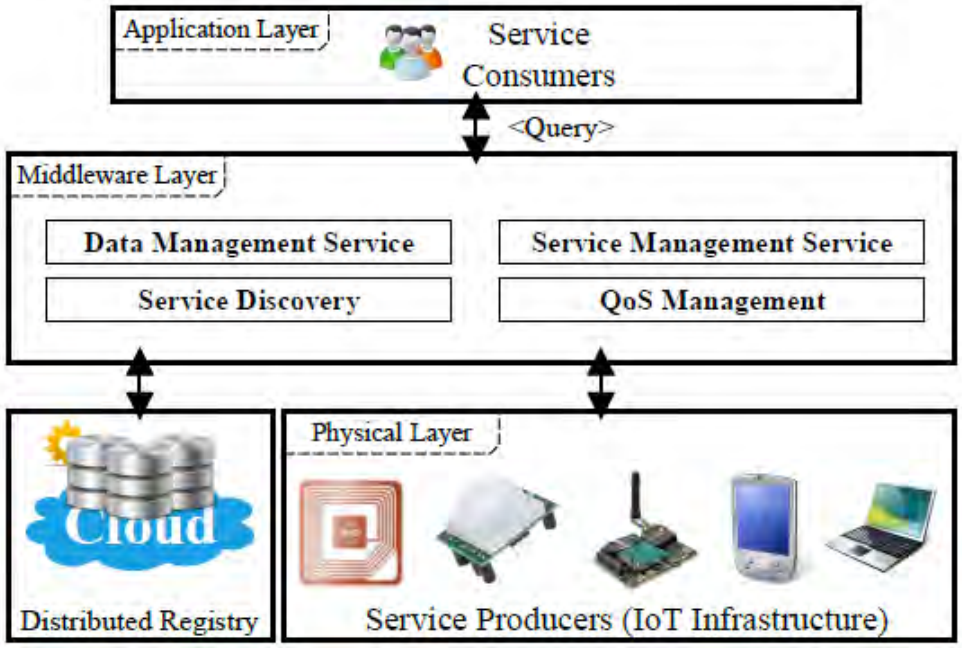

Figure 3.7: Modèle de conception générale pour un middleware orienté service

Des approches plus récentes ont tenté d'améliorer les fonctionnalités de la SOA et de l'adapter à l'IoT. Par exemple, SenseWrap fournit une interface de communication normalisée permettant de masquer aux applications les détails propres au capteur [162]. Il introduit le concept de capteur virtuel pour offrir une découverte transparente des capteurs. Cependant, la virtualisation n'est appliquée qu'aux capteurs et non aux actionneurs ou aux ressources informatiques, ce qui la rend totalement inadaptée aux environnements IoT. TinySOA utilise des mécanismes simples et déterministes pour l'enregistrement et la découverte des ressources WSN (par exemple, les nœuds de capteurs) [163]. Il ne prend en charge que quelques exigences fonctionnelles de base (par exemple, abstraction, découverte de ressources et gestion). CapteursMW [164] s'est révélé être un middleware adaptable et flexible pour la gestion des capteurs. Il permet une configuration simple et efficace des réseaux de capteurs sans fil pour la collecte d'informations. Toutefois, la reconfiguration peut échouer dans les applications critiques car elles définissent des règles de qualité de service strictes. MOSDEN supporte la détection en tant que modèle de service construit sur GSN [165]. Il repose sur une architecture de plug-in qui améliore l'évolutivité et la convivialité. Toutefois, les fonctionnalités prédéfinies de découverte de ressources / services et de mécanismes de composition de services présentées dans cette approche peuvent poser des problèmes dans un environnement IoT dynamique.

3.3.2.7 Middleware oriennté message (MoM)

Un middleware orienté message est utilisé depuis longtemps dans les communications réseau, en particulier dans les réseaux industriels tels que les systèmes de fabrication intégrés [166] [167]. Il offre une architecture basée sur les événements et un modèle de communication publication / abonnement. Dans une architecture basée sur des événements, les composants, les applications et tous les autres participants interagissent via des événements. Le modèle de publication / abonnement est un modèle d'interaction composé à la fois d'éditeurs et d'abonnés. Les sources de données (éditeurs) et les destinations (abonnés) sont découplées les unes des autres et les objets de données (messages) sont filtrés et livrés aux destinataires en fonction de sujets prédéfinis exprimés en abonnements grâce à un composant dédié appelé courtier de messages, comme illustré à la Figure 3.2. Le courtier peut être considéré comme un médiateur responsable de la gestion de la distribution des messages afin de fournir la bonne information au bon consommateur. La force de ce middleware réside principalement dans sa prise en charge de la communication asynchrone permettant un couplage lâche entre l'expéditeur et le destinataire.

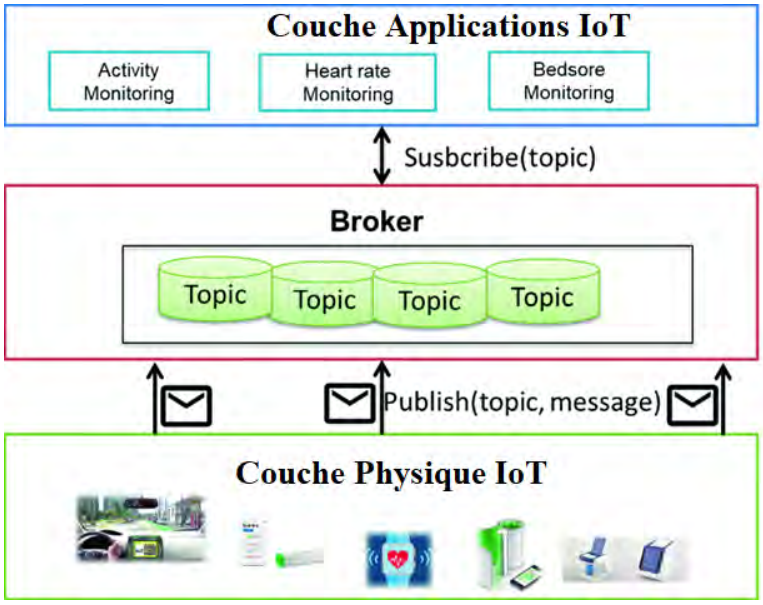

Figure 3.8: Modèle de conception général pour un middleware basé sur des événements

Depuis l'apparition de l'Internet des objets, le mécanisme de publication / abonnement a été mis en lumière pour son efficacité à offrir un couplage lâche. Comparé à l'architecture SOA qui est également largement proposée pour les solutions IoT, un MOM suit un modèle de distribution basé sur un message qui se concentre sur les informations. Il diffère du paradigme

classique client / serveur en ce que ni la source ni la destination du message ne doivent être connues l'une de l'autre avant la communication.

Peu de projets IoT ont proposé des solutions de publication / abonnement dans la littérature. Par exemple, le projet CenceMe a pour objectif de déduire automatiquement l'activité des personnes (par exemple, danser dans la soirée) sur la base d'un smartphone doté de capteurs, afin de la partager via les médias sociaux tels que Facebook [168]. Pogo [169], une infrastructure middleware de publication / abonnement pour la détection de téléphones mobiles, est un autre exemple permettant un accès facile aux données de capteurs sur les téléphones mobiles. Il utilise de simples abonnements thématiques pour gérer l'accès aux données des capteurs et signale des gains d'énergie significatifs grâce au filtrage par thème des données détectées sur les appareils mobiles.

D'autre part, un certain nombre de services basés sur le cloud dédié au stockage de données basées sur des capteurs sont désormais disponibles. Quelques exemples pouvant être mentionnés sont Xively, ThingSpeak et iDigi, qui prennent en charge les connexions utilisant le protocole MQTT (Message Queuing Telemetry Transport). Ils représentent une infrastructure évolutive qui permet aux utilisateurs de créer des produits et des services IoT, ainsi que de stocker, partager et découvrir des capteurs en temps réel. En résumé, un tableau comparatif est présenté dans le tableau 3.1. Dans la section suivante, nous décrivons les solutions middleware développées dans le contexte de la santé.

3.3.3 Solutions Middleware dans le secteur de la santé

Les soins de santé sont un sujet de recherche actif dans l'IoT où les applications peuvent être classées en AAL (Ambient Assisted Living). Les applications médicales sont principalement installées dans les hôpitaux et les maisons de retraite pour prévenir, détecter des maladies ou même surveiller l'état de santé des patients. Les applications AAL sont principalement installées chez le patient pour surveiller et suivre son état de santé et son activité à la maison. Dans ce contexte, des solutions middleware ont été proposées pour répondre aux exigences des domaines de l'IoT et de la santé. Sur le tableau 3.1, nous avons comparé les différents middlewares vus jusqu'à présent.

Tableau 3.1: Tableau comparatif des approches Middleware

Défis IoT		Approches middleware						
		Spécifique à l'application	Basé sur l'agent	Basé sur la VM	Tuples-espaces	Orienté base de données	SOA	MoM
Défis d'infrastructure	Interopérabilité technique		X	X			X	X
	L'évolutivité							X
	Interactions spontanées			X	X			X
	Infrastructure non fixée		X	X	X		X	X
	Abstraction	X	X	X	X	X	X	X
Défis d'application	Disponibilité	X	X	X	X	X	X	X
	Fiabilité		X	X		X	X	X
	Temps réél						X	X
	Confidentialité et sécurité							

3.3.4 Middleware SIP

Le middleware SIP est un mécanisme permettant de transmettre des données de service de périphérique via SIP. Le format de données du service de périphérique peut être divisé en trois types: données textuelles, signalisation de contrôle et flux multimédia. Par exemple, les informations d'état du module de transmission de données appartiennent aux données texte, le paquet RTSP du module de gestion multimédia appartient à la signalisation de contrôle et le flux vidéo du module de gestion multimédia appartient au flux multimédia. De même, la pile de protocoles SIP de la passerelle divise les messages SIP en quatre modules fonctionnels en fonction de différentes applications SIP. Le premier est le module d'enregistrement, qui est une implémentation de la méthode REGISTER pour recevoir des informations de périphérique et les enregistrer auprès du centre de service de périphérique. Le suivant est le module d'abonnement et de notification, qui peut implémenter les méthodes SUBSCRIBE et NOTIFY, accepter les abonnements d'utilisateur et envoyer des notifications aux utilisateurs abonnés lorsqu'un événement est détecté. Le troisième est le module d'invitation et de finition, qui est responsable de la mise en œuvre des méthodes INVITE et BYE, avec la réception d'une demande de session multimédia par des utilisateurs distants et la transmission d'informations de session et multimédia à la passerelle. Le dernier est le sous-module de diffusion INFO, qui est responsable de la mise en œuvre de la méthode INFO pour transmettre la signalisation de contrôle et les flux de média pendant la session. Différents modules de fonction et méthodes SIP peuvent gérer différents formats de message et différents messages de format en fonction de différentes fonctions de service de périphérique. Le middleware SIP pour l'interaction de

service de périphérique avec les données de communication SIP est conçu sur la base de cette relation.

Afin de réaliser les communications entre les deux programmes, le mécanisme de communication Socket est adopté. Un ensemble de modules de traitement de distribution de messages est implémenté entre le module de fonctions de la pile de protocoles SIP et le service de périphérique. Le serveur de socket est implémenté dans le module de traitement de la distribution des messages à proximité du service de périphérique. Et le client de socket est implémenté dans le module de traitement de la distribution des messages à proximité de la pile de protocoles SIP. Le module de traitement de la distribution du message détermine le type du message reçu et le distribue au module correspondant. La Figure 3.9 montre la structure détaillée du middleware SIP, où register / xml, vidéo / données, etc. sont les informations du champ Content-Type de l'en-tête de message SIP. Le schéma de conception offre une excellente évolutivité du système, un accès flexible aux modules et un couplage souple satisfaisant entre les modules de la couche application, qui peuvent parfaitement correspondre aux idées de programmation et aux fonctionnalités orientées objet de la conception modulaire.

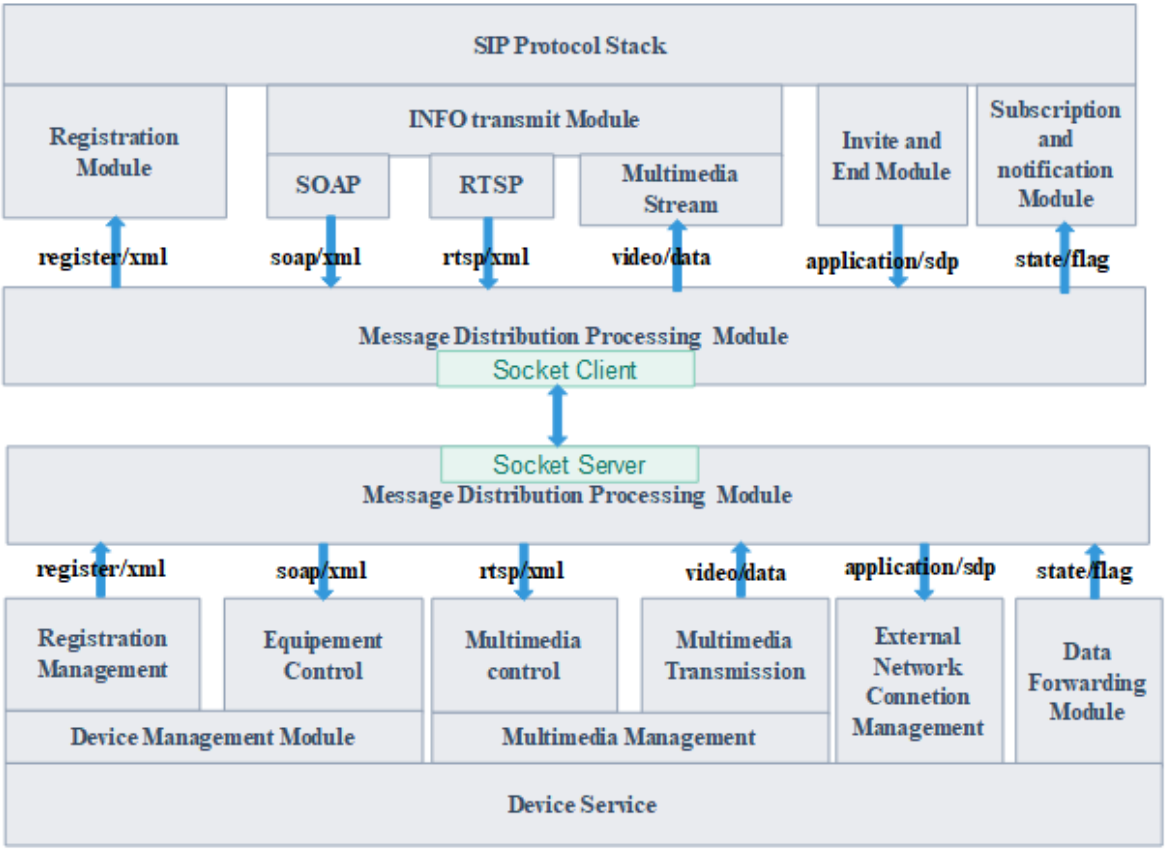

Figure 3.9: Structure middleware SIP

3.3.4.1 Conception de la base de donnée du Middleware SIP

L'accès à la base de données utilise les capacités de gestion des ressources du système d'une ville intelligente. L'accès des applications à la base de données peut être divisé en plusieurs étapes: la connexion à la base de données, la sélection de la base de données, l'opération de base de données et la fermeture de la connexion à la base de données. Habituellement, lorsque nous accédons à la base de données, la connexion et la déconnexion prennent moins de temps et occupent trop peu de ressources système. Cependant, lorsque le système accède plus fréquemment à la base de données, la consommation de temps de ces deux étapes devient le facteur d'influence principal. Sur cette base, la passerelle doit être conçue avec un middleware de base de données en utilisant la technologie de pool de connexions de base de données.

Dans le système Linux, MySQL ne fournit que la fonction API originale pour la prise en charge du langage C, et ces fonctions API sont très complexes. Le pool de connexions de base de données ré-encapsule ces API. Le middleware de base de données utilise cette technologie pour optimiser la connexion à la base de données. Il est conçu pour stocker une collection de bases de données. La figure 3.10 est un schéma de principe de la conception d'un middleware de base de données. Cette conception élimine le besoin de créer une connexion de base de données et de fermer la connexion à la base de données, ce qui améliore considérablement le nombre de visites et réduit la consommation de ressources système.

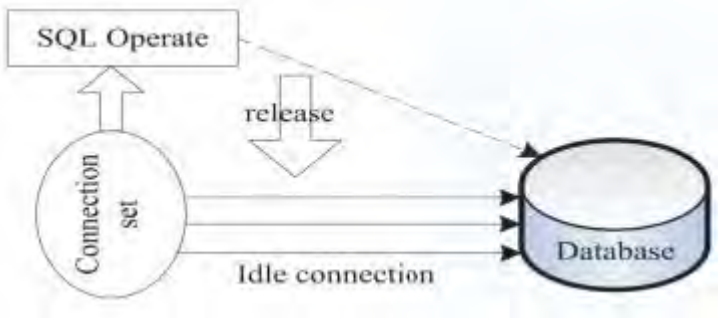

Figure 3.10: Principe de conception du middleware de base de données SIP

3.3.5 Entités middleware et sources de trafic

L'élaboration des mécanismes de gestion de la QoS doit tenir compte des caractéristiques des entités intervenantes. Les premières entités sont les entités Middleware qui, en fonction de leur performance, peuvent induire une dégradation de la QoS. La gestion et l'intégration de ces mécanismes se fait donc au niveau de ces entités ou bien en amont. Les deuxièmes entités sont

les sources de trafic. Ce sont elles qui peuvent être critiques et exprimer des besoins en QoS. Dans cette section, nous décrivons ces deux types d'entités.

3.3.5.1 Entités middleware

Le modèle contextuel du système IoT considéré (Figure 3.11) se base sur des entités Middleware intermédiaires entre les applications et les équipements contribuant au traitement du trafic [17]. Ce trafic est appelé de manière générique trafic WTP (Web Transfer Protocol) indépendamment du protocole de transfert (par exemple, HTTP ou CoAP). Une gateway IoT est une passerelle permettant d'interconnecter les équipements IoT (par exemple, capteurs, actionneurs) au reste du système. Une gateway est contrainte d'être au même emplacement physique que ces équipements. Elle est (dans notre contexte) déployée sur une plate-forme physique et donc limitée en terme de ressources informatiques. Un serveur IoT constitue le point de concentration permettant l'interconnexion des différentes applications IoT avec les différentes gateway pour l'accès aux équipements IoT. Un serveur peut être déployé sur un environnement virtualisé tel que le cloud.

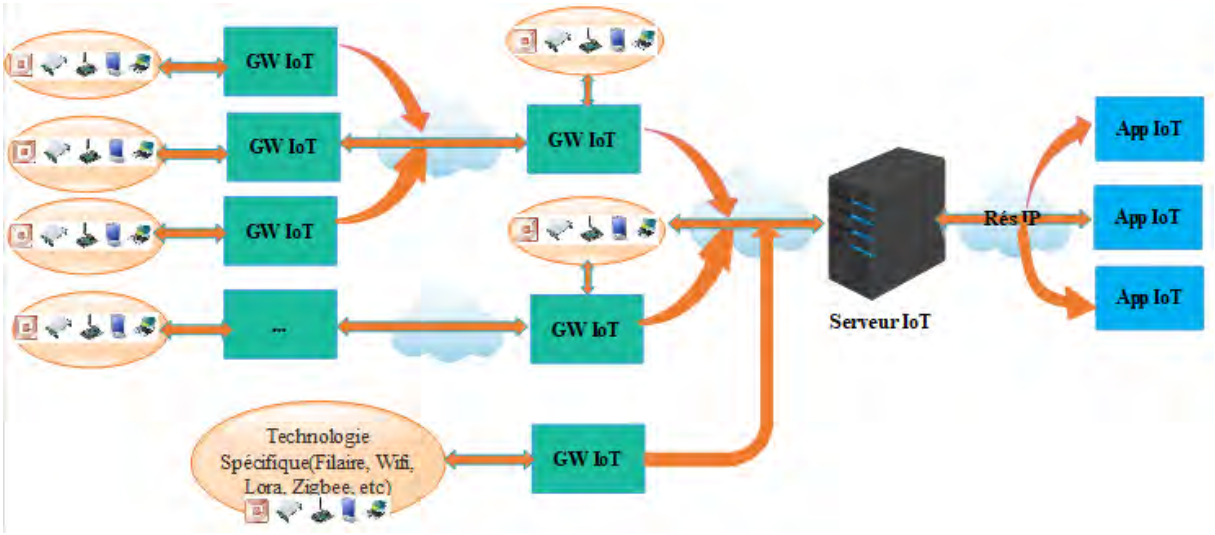

Figure 3.11: Modèles contextuel du système IoT

En fonction de leur état, ces entités sont susceptibles de conduire à la non satisfaction des besoins en QoS du trafic issu des sources. C'est donc au profit de ces deux entités que les mécanismes (en tout ou partie, et en fonction de l'entité) vont être déployés afin de respecter les contraintes QoS des sources tout en assurant la délivrance des services IoT.

La gestion de ces entités Middleware se passe par la proposition de mécanismes orientés QoS. Dans notre approche, cette gestion est assurée de manière dynamique et auto-adaptative suivant le paradigme de l'Autonomic Computing proposé par IBM [170]. Ce paradigme

propose une boucle MAPE-K d'autogestion enchaînant des phases de supervision (M-Monitoring) de l'entité gérée (entité Middleware), d'analyse (A-Analysis) des causes de dégradation, de planification (P - Planning) et d'exécution (E-Execution). Ce paradigme met en œuvre des politiques, des règles, des modèles et/ou des algorithmes maintenus dans une base de connaissance (K).

3.3.5.2 Sources de trafic

Les sources de trafic sont les entités génératrices de trafic. En fonction du service métier fourni, le trafic émanant de ces sources peut être critique et présenté des besoins en QoS (délai, pertes, etc.). Le système doit donc répondre à ces besoins. Nous pouvons distinguer ces sources en fonction de l'entité Middleware destinataire (Tableau 3.2), c'est-à-dire via laquelle passe le trafic.

Tableau 3.2: Sources du trafic en fonction du Middleware destinataire

Middleware destinataire	Source de trafic
Serveur IoT	Application IoT
	Gateway IoT
	Equipement IoT implémentant la couche de service Middleware
Gateway IoT	Application IoT
	Serveur IoT
	Equipement IoT non capable d'implémenter la couche de service

Vis-à-vis du besoin en QoS, nous distinguons deux types de sources :

- **source QoS-unaware :** source qui n′est pas dotée de moyens pour exprimer ses besoins en QoS. Le système IoT doit donc, en fonction des caractéristiques de la requête (source, destination, méthode, etc.), identifier la priorité à lui accorder pour son traitement ;

- **source QoS-aware :** contrairement au premier type, les sources de ce type sont capables d′ exprimer leurs besoins en QoS au système, et éventuellement de participer à la gestion de la QoS en interagissant avec le système IoT, par exemple, en adaptant le débit de leurs requêtes.

Les types de mécanismes que nous considérons dépendent du type de l′entité Middleware gérée. Une gateway, qui est déployée sur une machine physique, ne peut tirer parti que des mécanismes orientés trafic. Un serveur IoT, déployable / déployé dans un environnement cloud, peut exploiter les mécanismes orientés ressources en plus de ceux orientés trafic.

Dans la suite de ce chapitre, nous définissons ces deux types de mécanismes, leur instanciation dans le contexte considéré, ainsi que leur validation dans le cadre d′un certain nombre de scénarios.

3.3.6 Mécanismes de gestion orientés trafic

Les mécanismes de gestion orientés trafic sont inspirés de ceux utilisés au niveau des couches Transport et IP de l'architecture de l'Internet pour la gestion de la QoS. Ces mécanismes se positionnent au niveau Middleware et sont, dans notre travail, placés en amont de l'entité Middleware. Ils agissent directement sur le trafic WTP (Web Transfer Protocol) entrant. Dans la gestion proposée, nous nous basons sur un principe de différenciation de service analogue à celui proposé dans le modèle DiffServ de l'IETF [171]. Nous distinguons deux composants essentiels : le composant de classification et de marquage qui permet d'attribuer des priorités au trafic WTP, et le proxy orienté priorité qui différencie le traitement du trafic en fonction de sa priorité.

3.3.6.1 Classification et Marquage du trafic

Inspirée de la terminologie DiffServ, la fonction de classification et de marquage fournit la base pour la mise en œuvre des techniques de gestion de la QoS. Cette fonction se base sur la notion de priorité. Ces priorités correspondent au marquage du trafic en fonction de sa classe d'appartenance. La classification du trafic WTP se base sur des critères tels que l'adresse de la source, le type de la ressource destinataire, etc.

- **En-têtes WTP**

Le composant de classification et de marquage se base sur certains des champs d′en-tête du trafic pour d′abord le classer et ensuite lui attribuer une priorité. Par défaut, les protocoles WTP (HTTP ou CoAP par exemple) ne comportent pas de champ permettant de spécifier une

priorité. Ceci étant, ils permettent d′ajouter des en-têtes personnalisés. Nous exploitons donc cette propriété en intégrant un en-tête que nous appelons TOS_WTP (*Type of Service for WTP*), analogue au champ TOS des paquets IP.

Afin de garantir que l'en-tête TOS_WTP soit toujours à jour pour les sources QoS aware, nous ajoutons un autre en-tête TOS_WTP_V qui reflète sa version. En cas de mise à jour des politiques de classification et de marquage, le TOS_WTP_V sert à vérifier si le TOS_WTP est à jour avec la nouvelle politique. S'il ne l'est pas, l'en-tête TOS_WTP ne sera pas pris en considération et le trafic sera traité comme un trafic émanant d'une source QoS-unaware. Suite à cette action, une notification est envoyée à la source émettrice de ce trafic pour l'avertir.

❖ **Classification du trafic**

L'objectif de cette étape est de faire une distinction entre les différents trafics WTP. En fonction des caractéristiques du trafic, une classe lui sera attribuée. Cette classification concerne le trafic en provenance des sources QoS-unaware qui ne sont pas capables d'ajouter directement l'entête TOS_WTP pour exprimer leurs besoins en QoS.

Usuellement, un trafic applicatif est souvent défini par cinq éléments : adresse IP de la source, adresse IP destination, port source, port destination et protocole de Transport. Dans notre cas, nous élaborons la classification de façon plus complexe sur la base des champs suivants (Tableau 2.2) :

- **Adresse source** : si c'est source est une application, étant donné que sur une même machine, plusieurs applications peuvent y être hébergées, la classification se fait sur la base de son adresse IP et son numéro de port. Par contre, si la source est l'entité Middleware (serveur ou gateway), la classification se fait seulement sur la base de l'adresse IP ;
- **Adresse destination :** idem que dans le cas précédent à ceci près que la classification se base sur l′adresse de la destination au lieu de celle de la source ;
- **Type de source :** la distinction du trafic WTP est faite sur la base du type de source : application IoT, serveur IoT, gateway IoT ou équipement ;
- **Type de destination :** la distinction du trafic WTP est faite sur la base du type de destinataire : application IoT, serveur IoT (base de données locale), gateway IoT (base de données locale) ou équipement ;
- **Type de données :** la distinction est faite en fonction du type de trafic entrant : données brutes, audio temps réel, vidéo temps réel, image, etc. ;

- **Type d′interaction :** la distinction est faite entre les données de signalisation (requête, souscription, etc.) et les données utiles (réponse, notification, données périodiques, alarme, etc.) ;

- **Méthode d′interaction :** la distinction est faite à partir des méthodes WTP utilisées : création, récupération, mise à jour, exécution ou suppression.

Tableau 3.3: Critères de classification du trafic

Base	**@SRC ou @DST**		**Type SRC ou DST**	**Type de données**	**Type d'interaction**	**Méthode d'interaction**
Critère(s) de classification	**APP**	**MW**	Contexte de la requête WTP (APP, SRV, ou GW)	Brutes Image Audio Vidéo	Signalisation ou donnée utile	GET POST PUT DELETE
	@IP+n° port	@IP (Srv ou GW)				

A l'issue de cette classification, chaque requête est ainsi affectée à une classe donnée. A titre d'exemple, pour une classification sur la base du type de trafic, nous pourrions avoir les classes illustrées par le Tableau 3.4.

Tableau 3.4: Exemple de politique de classification de trafic basée sur le type de données

Type de données	Classe
Brutes	Classe 1
Vidéo	Classe 2
Audio	Classe 3
Images	Classe 4

Cette classe est ensuite utilisée par le composant de marquage afin d'attribuer au trafic la priorité correspondante via l'en-tête TOS_WTP.

Marquage du trafic

Cette étape consiste en l'attribution d'une priorité à chaque trafic en fonction de sa classe d'appartenance. Le marquage se base sur une correspondance entre les classes et les priorités fournies en amont par le gestionnaire autonomique (Tableau 3.5). L'en-tête TOS_WTP est ajouté et se voit attribuer la valeur correspondant à la priorité du trafic.

Tableau 3.5: Exemple de politique de marquage de trafic

Classe	Priorité
Classe 1	PRIOR_1
Classe 2	PRIOR_2
Classe 3	PRIOR_3
Classe 4	PRIOR_4

Soit l'exemple du domaine de la télésurveillance d'un patient à domicile. Dans ce domaine, nous considérons trois applications de criticités différentes (hautement critique, moyennement critique et non critique) :

- L'application de télésurveillance postopératoire du patient est la plus critique. Elle est à la fois très sensible au délai (inférieur à 300 ms [172]) et aux pertes (aucune perte) ;
- L'application du suivi de la pression artérielle est moyennement critique (sensibilité moyenne au délai (10 s) et aux pertes (50%)) ;
- L'application du suivi du pourcentage de calorie n'est pas critique.

Le Tableau 3.6 suivant illustre les priorités attribuées au trafic (http par exemple) échangé dans le cadre des trois classes d'applications IoT pour la satisfaction de leurs besoins en délai ou en pertes.

Tableau 3.6: Exemple d'attribution de priorités selon la sensibilité du trafic

Application IoT de télésurveillance	Délai seuil	Pertes Seuil	TOS_WTP (besoin en délai)	TOS_WTP(besoin en pertes)
Postopératoire	300ms	0%	PRIORITY_DELAY_HIGH	PRIORITY_LOSSES_HIGH
Pression artérielle	10s	50%	PRIORITY_DELAY_MEDIUM	PRIORITY_LOSSES_MEDIUM
Calories Brulées	-	-	PRIORITY_DELAY_LOW	PRIORITY_LOSSES_LOW

3.3.6.2 Architecture du CCM

Le composant de classification et de marquage (CCM) constitue le premier élément par lequel transite le trafic. Ce composant offre une interface de gestion permettant de l'activer, de le désactiver, ou de le configurer en termes de politiques de classification et de marquage.

La Figure 3.12 représente l'architecture fonctionnelle du CCM. Elle est constituée des composants suivants :

- **Gestionnaire CCM** : reçoit l'ordre d'activation / désactivation de la part de l'Autonomic Manager ainsi que les politiques pour guider la classification et de marquage ;
- **Récepteur WTP :** responsable de la réception du trafic WTP issue de la source. Ce récepteur est capable de recevoir le trafic quel que soit le protocole utilisé (HTTP ou
- CoAP par exemple) ;
- **Contrôleur CCM :** reçoit le trafic de la part du Récepteur WTP. En cas d'activation, il dirige le trafic soit vers le Classificateur WTP si l'en-tête TOS_WTP n'existe pas, sinon directement vers le Client WTP si l'en-tête existe déjà. En cas de désactivation, le trafic est directement dirigé vers le Client WTP ;
- **Classificateur WTP :** classifie le trafic entrant en fonction de la politique de classification. Ce composant réalise tout d'abord l'analyse du trafic en fonction de la politique de classification (adresse IP source, adresse IP destination, l'URL de la ressource, la méthode, etc.). Ensuite, il identifie la classe d'appartenance du trafic en fonction de ses caractéristiques. Il redirige ensuite le trafic vers le composant Marqueur de priorité ;
- **Marqueur WTP :** attribue une priorité pour chaque classe en se basant sur la politique communiquée par le Gestionnaire de composants. Il ajoute l'en-tête TOS_WTP au trafic et lui attribue une valeur correspondant à la priorité du trafic ;
- **Client WTP :** redirige le trafic vers la prochaine entité. Cette dernière peut être soit le Proxy orienté priorités, soit le système destinataire. L'adresse de redirection est communiquée par le Gestionnaire Autonomique.

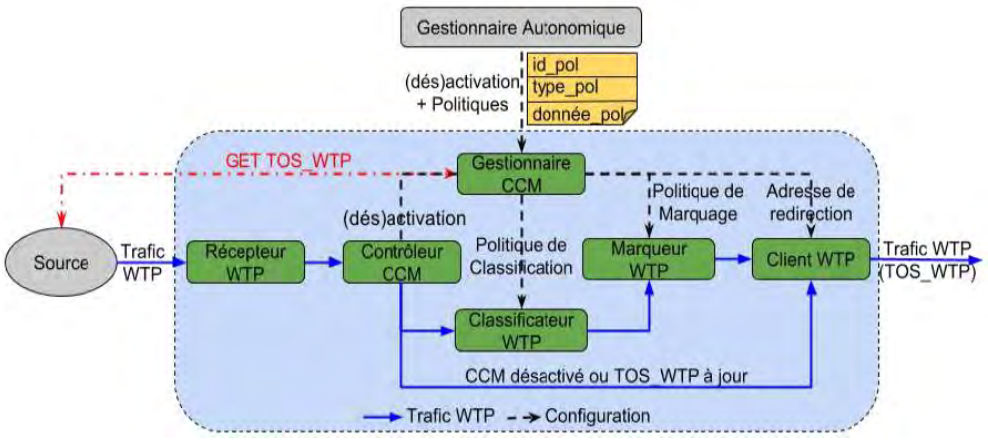

Figure 3.12: Architecture fonctionnelle du CCM

La procédure d'ajout de l'en-tête TOS_WTP contenant une priorité est illustrée comme suit (Figure 3.13) :

- initialement, le CCM est en état de désactivation. Le trafic passe par le composant Récepteur WTP ensuite par le Contrôleur CCM" qui le redirige directement vers le Client WTP ;
- le CCM est activé par le Gestionnaire Autonomique. Ce dernier lui communique la politique de classification au Classificateur WTP et la politique de marquage au Marqueur WTP ;
- à chaque réception d'un trafic WTP, le Récepteur WTP l'oriente vers le Contrôleur CCM qui vérifie l'existence de l'en-tête TOS_WTP et la validité du TOS_WTP_V. Si c'est le cas, il l'oriente directement vers le Client WTP. Sinon, il le dirige vers le Classificateur WTP qui le classe selon sa politique de classification et l'oriente vers le Marqueur WTP ;
- le Marqueur WTP effectue la correspondance entre la classe et la priorité et ajoute l'entête TOS_WTP avec la priorité correspondante ;
- enfin, le trafic est acheminé vers le Client WTP pour le rediriger. L'adresse de redirection est fournie par l'AM.

Les sources QoS-aware peuvent communiquer via une autre interface avec le Gestionnaire CCM pour récupérer les politiques de classification et de marquage existantes. Elles pourront donc ajouter par elles-mêmes l'en-tête TOS_WTP de leurs trafics WTP en fonction de leurs besoins en QoS.

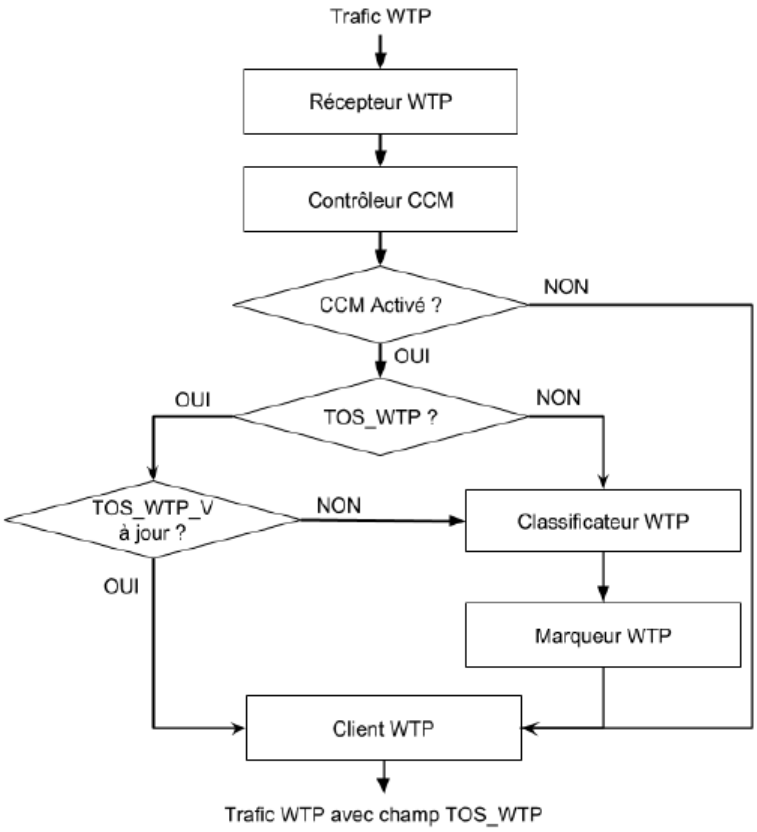

Figure 3.13: Organigramme de programmation du CCM

3.4 CONCLUSION

Dans ce chapitre, nous avons présenté les middlewares dans l'internet des objets, en définissant d'abord les concepts théoriques y relatifs.

Puis, nous avons présenté les modèles technologiques que nous utiliserons dans la suite de la thèse. Nous avons traité des modèles de middleware de trafics et identifié le modèle qui convient dans le e-santé, e-learning et e-agriculture.

Ensuite, nous avons présenté le principal type de mécanisme pour la gestion de la QoS, orienté trafic qui est adapté aux différents environnements de déploiement, physique ou virtuel. Les mécanismes orientés trafic sont inspirés des approches utilisées au niveau des couches IP et Transport de l'architecture de l'Internet, notamment l'approche DiffServ. Ces mécanismes permettent de gérer le trafic en fonction de sa priorité. Ils sont basés sur deux composants placés en amont du Middleware destinataire. Le composant de classification et de marquage (CCM) permet de classer et de marquer le trafic avec une priorité donnée en fonction de plusieurs critères.

4 Contribution à l'amélioration de la fourniture de services médicaux à distance aux patients vivant en zones rurales

4.1 Introduction

Ce chapitre propose un nouveau modèle de middleware IoT intégrant le WebRTC et WoT, afin de fournir des services de communication en temps réel enrichis de données contextuelles recueillies à partir des environnements des utilisateurs. Une application de ce modèle est ensuite démontrée dans le domaine du e-Santé, principalement pour la fourniture de services médicaux à distance aux patients, en particulier ceux vivant dans les zones rurales.

Ensuite, pour garantir la confidentialité et l'intégrité des données personnelles, un modèle de contrôle d'accès et de délégation de rôles est proposé. Ce modèle en prend en compte le contexte, le lieu, le temps, la durée pour faciliter la délégation de rôles. Enfin, nous proposons un moyen de fiabilisation de la base de présence des plateformes utilisant le protocole de signalisation SIP.

4.2 Architecture d'un système de mutualisation et de partage d'informations via un middleware IoT

L'architecture proposée vise entre autres à assurer les mêmes interactions que dans un middleware orienté service traditionnel ou bien par web service. Pour cela, nous prenons en considération les critères de conception d'environnements informatiques qui permettent d'assurer une efficacité d'interaction entre le patient et le personnel soignant afin de faciliter l'accès à des soins sanitaires dans le milieu rural. Ce modèle est conçu pour supporter les fonctions suivantes :

- L'interactivité ;
- La disponibilité des services à tout moment ;
- La fourniture en temps réel d'information des capteurs ;
- La sécurité et la confidentialité des données du patient.

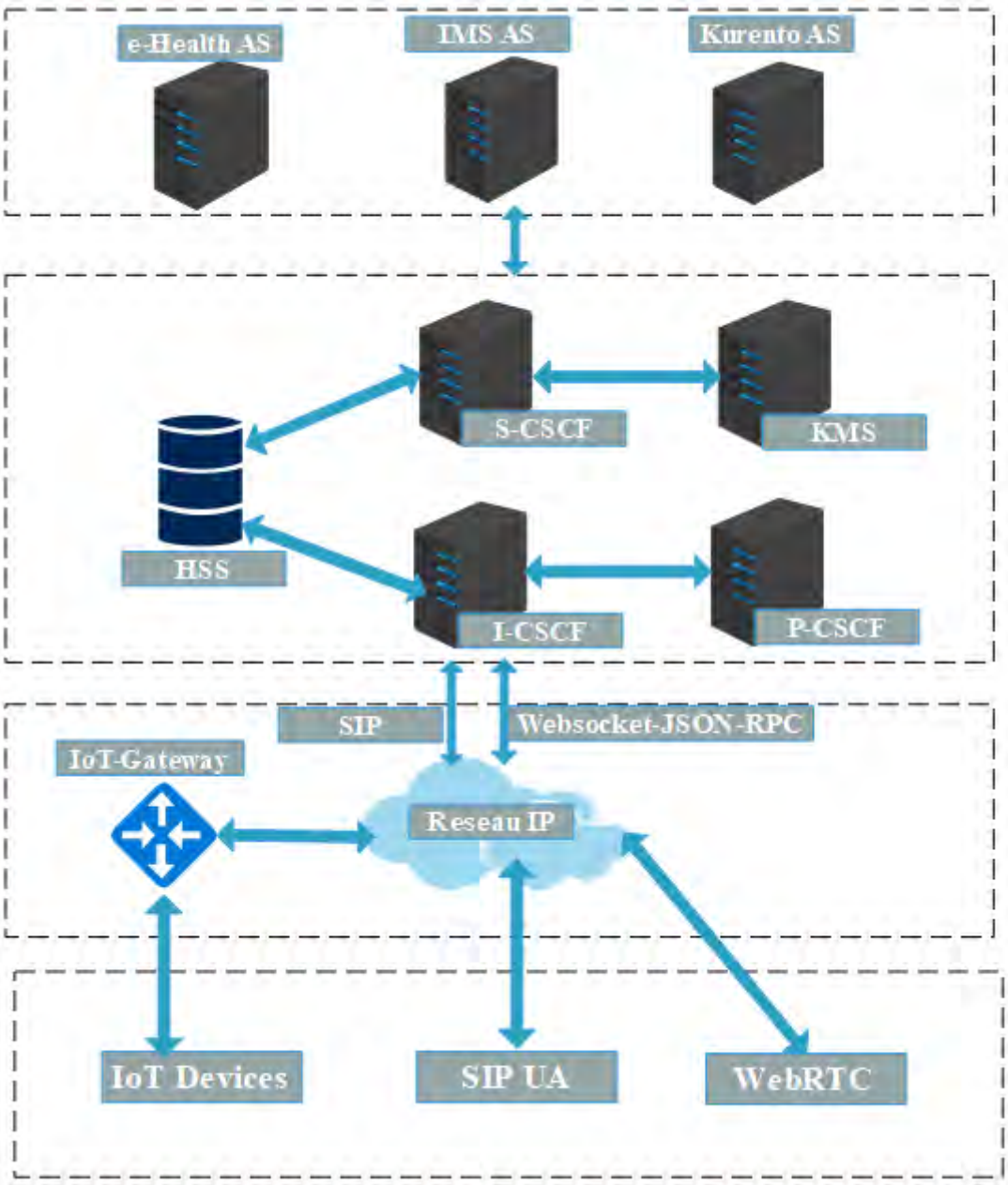

Figure 4.1: Architecture d'un middleware permettant le couplage WebRTC, IMS et WoT

Notre approche contribue à la mise en place d'un système de mutualisation et de partage d'informations via un middleware IoT pour faciliter l'accès à des soins de qualité aux populations vivant en zone rurale.

Chaque poste/ou district sanitaire est doté d'un système de capteurs biométriques et d'un ensemble de dispositif informatique permettant aux personnels soignants de choisir le meilleur spécialiste suivant les catégories de maladies dont souffrent le patient sans quitter sa localité. Cette architecture offre donc aux malades vivant en zone rurale la possibilité :

- D'être consulter par les meilleurs médecins spécialistes sans quitter sa localité ;

- De réduire les factures d'hospitalisation ;
- D'anticiper sur les éventuels risques de maladies ;
- D'accéder à des soins de santé de qualité tout en restant dans sa localité.

Dans la suite, une abstraction de notre architecture est présentée, où sont définis les différents éléments impliqués ainsi que la manière dont ils communiquent entre eux. La Fig. 4.1 fournit les interactions principales et distingue quatre composants principaux :

1) Le « Client WebRTC » situé dans le navigateur, le client SIP et les « objets » de l'internet des objets comprennent une large gamme de terminaux capable d'envoyer et recevoir des informations ;
2) La couche accès réseau et passerelle ;
3) La couche session constituée par le réseau IMS, permet aux utilisateurs finaux d'utiliser le protocole de Signalisation SIP pour accéder à certains services et au serveur média Kurento ;
4) Et enfin la couche application qui fournit les services via les serveurs d'application pour la santé, pour le réseau IMS et celle multimédia.

Le client WebRTC peut communiquer directement avec les objets intelligents s'ils disposent de suffisamment de fonctionnalités pour établir un canal (HTTP ou autres), ou avec la passerelle qui servira d'intermédiaire entre les utilisateurs et les périphériques sous contrainte. La passerelle est capable de communiquer avec n'importe quel capteur / actionneur, directement ou via un contrôleur qui peut être un Arduino, un Raspberry Pi, etc. La passerelle est dotée de plusieurs piles lui permettant d'utiliser diverses technologies de communication, telles que WIFI, Bluetooth, Zigbee, LoRa, etc.

Le système interagit également avec un serveur de bases de données pour la gestion de stratégies de contrôle d'accès. En outre, l'architecture présentée peut être affinée en ce qui concerne la possibilité d'un traitement intensif des données à la périphérie du réseau à l'aide des paradigmes Fog Computing qui est un terme introduit par Cisco en 2013 pour décrire un regroupement de ressources de calcul et réseau destinées à prendre en charge (sans s'y limiter) l'Internet des Objets (IoT) ou Mobile Edge Computing (MEC). Il aide à résoudre plusieurs problèmes liés au cloud tels que la latence élevée, la connaissance de la localisation, la fiabilité, etc. Le déplacement des données vers le meilleur emplacement pour le traitement peut être résolu par Fog Computing selon [173] [174] [175].

Cette solution est pertinente pour les applications IoT, telles que les réseaux intelligents et les maisons intelligentes, qui se caractérisent par leur sensibilité au temps de latence et nécessitent donc une analyse immédiate des données et la prise de décision pour la conduite de l'action.

Dans notre architecture, les données recueillies des capteurs IoT sont transférées directement en temps réel via WebRTC sans aucun traitement, d'où l'omission de l'utilisation du calcul par brouillard à la périphérie du réseau. Cependant, l'importance d'une telle approche n'est pas négligeable et peut être explorée pour des travaux futurs. Le traitement des données IoT sur le bord peut fournir des réponses rapides, en particulier en ce qui concerne les données anormales sensibles, telles qu'une forte fièvre d'un patient recueillie par un capteur de température, qui nécessite une réaction rapide.

4.3 Présentation des outils et technologies utilisés pour les contributions

Dans cette section nous allons présenter les outils et technologies qui nous ont permis à proposer des solutions innovantes dans le domaine de la télémédecine, afin que les populations rurales accèdent aux services de soins de qualité. Ces solutions sont basées sur les technologies comme le serveur multimédia WebRTC le plus puissant du moment, Kurento, la plateforme Kamailio IMS, le Web of Thing. Pour gérer la sécurité nous avons proposé un modèle de contrôle d'accès basé sur la délégation et l'organisation (DORBAC) qui est une extension du modèle OrBAC.

4.3.1 Kurento Media Server

Les applications WebRTC traditionnelles sont normalisées, de sorte que les navigateurs puissent communiquer directement sans la médiation d'infrastructures tierce. Cela suffit pour offrir des services multimédias de base, mais des fonctionnalités telles que les communications de groupe, l'enregistrement de flux, la diffusion ou le transcodage sont difficiles à mettre en œuvre. Pour cette raison, les applications les plus intéressantes nécessitent l'utilisation d'un serveur multimédia.

Kurento est un serveur multimédia WebRTC open source qui permet de créer des applications de traitement de média basées sur le concept de pipelines. Les pipelines médias sont créés par des modules d'interconnexion appelés Media Elements. Chaque Media Element fournit une fonctionnalité spécifique. Kurento Media Server (KMS) contient des Media

Elements capables par exemple d'enregistrer et de mixer des flux, de faire de la vision par ordinateur, etc. Kurento Media Server offre les capacités de création de pipelines médias via un simple protocole réseau basé sur JSON-RPC. Cependant, pour simplifier davantage le travail des développeurs, une API client implémentant ce protocole et exploitant directement les Media Elements et les pipelines est fournie. Actuellement, l'API client Java et JavaScript est prête à l'emploi pour les développeurs [176]. En tenant compte des modules intégrés, la boîte à outils Kurento est détaillée sur la figure 4.2.

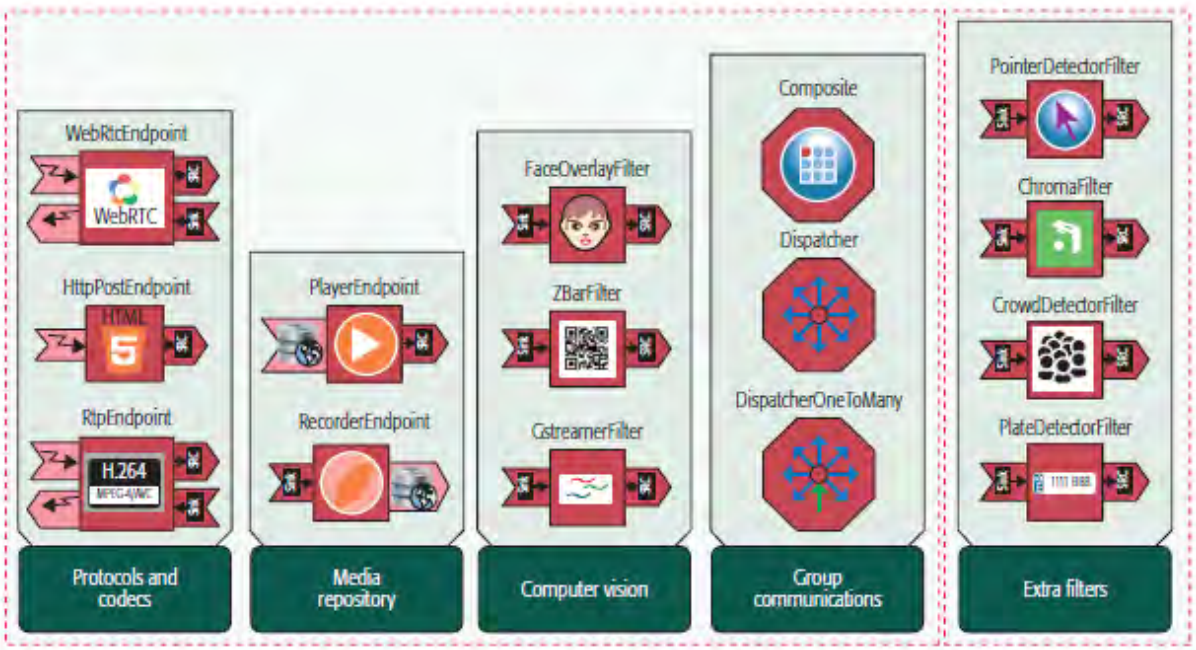

Figure 4.2: Kurento Media Elements toolbox

4.3.2 Kamailio et Module IMS

Kamailio est un projet initié par SER. Il s'agit d'un serveur SIP. A partir de la version 4, il a intégré les modules IMS ce qui offre la possibilité de l'utiliser comme plateforme pour bâtir un cœur de réseau IMS. Selon 3GPP, IMS est devenu la norme pour l'utilisation de la voix et d'autres services multimédias sur les réseaux LTE et d'autres réseaux IP de manière efficace, securisée et fiable. Il permet de combler le fossé entre les services vocaux à commutation de circuit et les technologies IP pour ouvrir la voie au development de services multimédia. IMS utilise le protocole SIP pour la gestion des multimédia. De nombreuses extensions SIP obligatoires sont intégrées pour gérer la sécurité et la fiabilité. Les différents modules Kamalio IMS sont principalement ims_auth, ims_isc, ims_icscf, ims_qos, ims_registrar_pcscf, ims_registrar_scscf, ims_usrloc_pcscf, ims_usrloc_scscf et

ims_ro_interface. La figure 4.3 nous montre les different modules et interaction avec les entités IMS.

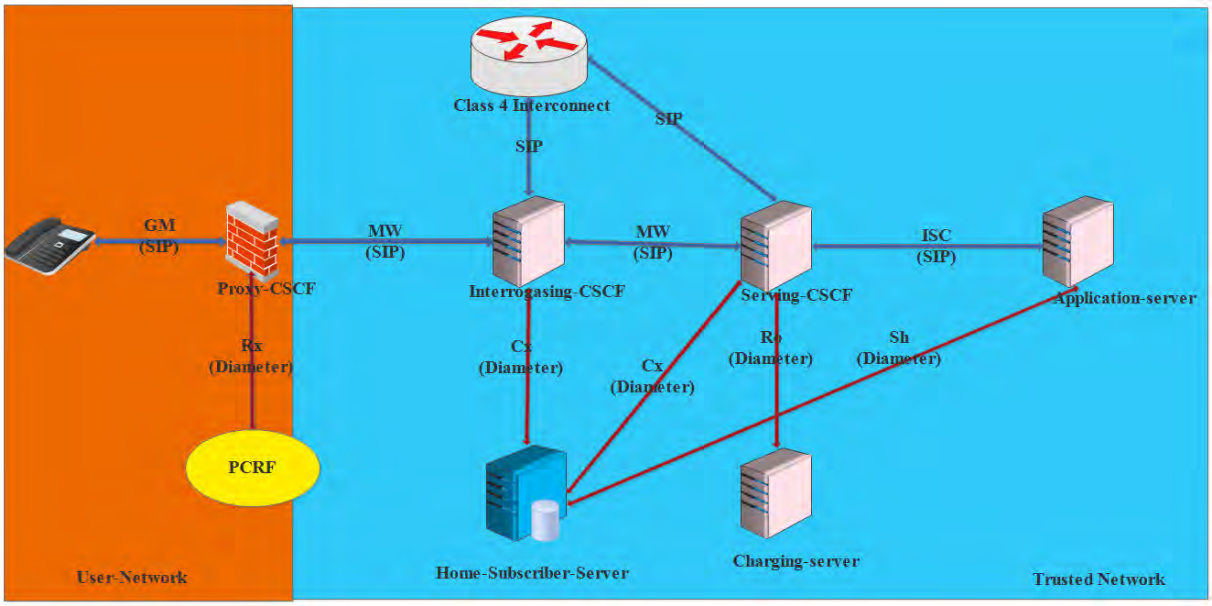

Figure 4.3: Architecture détaillée des entités et liaisons du réseau IMS bâtit à partir de Kamailio

Une explication détaillée des différents modules Kamailio IMS sont:

ims_auth : Ce module est utilisé pour l'authentification dans IMS. Dans l'IMS, la fonction S-CSCF n'est qu'un « intermédiaire » entre l'UE et le serveur HSS. Il communique via l'interface Cx avec le serveur HSS pour extraire les informations d'authentifications des abonnés. Actuellement, le module prend en charge les schémas d'authentification MD5 et AKAv1 / 2. Bien entendu, cela dépendra également des schémas d'authentification pris en charge par votre serveur HSS ;

ims_isc : Ce module implémente l'interface IMS Service Control entre le S-CSCF et le SIP Application Server (AS). Il utilise la logique requise au niveau de la fonction S-CSCF pour déterminer le SIP AS à appeler et à quel moment, en fonction des critères de filtre initiaux spécifiques à l'abonné. Il utilise une méthode au niveau de la fonction S-CSCF pour déterminer le SIP AS à interroger en fonction des critères de filtrages spécifiques à l'abonné ;

ims_icscf : Ce module implémente les fonctions requises pour créer un IMS I-CSCF. Il est déclenché à partir du fichier de configuration et peut interroger un HSS tiers pour trouver le S-CSCF concerné pour une demande ;

ims_qos : Ce module implémente les fonctions d'autorisation entre P-CSCF et PCRF. Le fichier de configuration déclenche le P-CSCF pour envoyer des demandes d'autorisation de Diameter au PCRF. Les appels de rappel CDP, Dialog et Usrloc sont utilisés pour signaler lorsqu'une boîte de dialogue est terminée, la session Diameter a pris fin ou l'utilisateur a été désenregistré ;

ims_registrar_pcscf : Ce module implémente simplement la logique de traitement REGISTER pour les messages REGISTER dans une configuration P-CSCF ;

ims_registrar_scscf : Ce module implémente simplement la logique de traitement REGISTER pour les messages REGISTER dans une configuration S-CSCF.

ims_usrloc_pcscf : Ce module est l'équivalent IMS du module de stockage kamailio usrloc. Il stocke les informations d'abonné IMS au P-CSCF. Ce module est requis parce que, au niveau du P-CSCF, un abonné peut se trouver dans de nombreux états différents (non enregistrés, enregistrés, etc.). De plus, il existe des métadonnées très différentes pour un abonné par rapport à SIP standard, par exemple, les informations IPSec, les informations de session Diameter Rx, plusieurs identités publiques (IMPU) par contact, etc ;

ims_usrloc_scscf : Semblable au module ims_usrloc_pcscf, ce module sert de module de stockage d'abonné pour ceux d'IMS au S-CSCF. Ce stockage comprend par exemple l'abonnement IMS téléchargé à partir du serveur HSS pour chaque abonné, les adresses de la fonction de taxation, etc ;

ims_ro_interface : Ce module implémente un mécanisme de facturation en ligne utilisant l'interface IMS Diameter Ro. L'interface Ro est destinée à être utilisée dans une configuration Kamailio S-CSCF et liée à un système de facturation en ligne (OCS). Actuellement, le module est purement basé sur le temps (c'est-à-dire que la charge de l'unité et les réservations sont en secondes) et utilise l'algorithme SCUR (Session-Charging-with-Unit-Reservation) (RFC 4006). Le module dépend du module dialog_ng pour suivre les «dialogues» à charger, ainsi que pour les supprime une fois que le crédit est épuisé ;

4.3.3 Le Web of Things

Le Web of Things (WoT) est une spécification de l'Internet des objets (IoT) fournissant une abstraction de connectivité à des objets intelligents. Sa particularité est l'existence d'une couche d'application standard basée sur les normes Web. Comme la plupart des spécifications

IoT, le WoT utilise différents protocoles de communications tels que MQTT, AMQP, CoAP. Ainsi un ensemble de capteurs utilisant un de ces protocoles peuvent constituer un groupe. Les principaux intérêts de l'utilisation du WoT au lieu de l'IoT classique résident dans la simplicité du développement grâce aux nombreuses API et l'accès via un navigateur. Une API de WoT peut être présente dans l'objet intelligent lui-même ou dans un proxy qui peut agir au nom de l'objet intelligent [71].

4.3.4 Modèles de contrôles d'accès

La plupart des architectures actuelles, notamment celles relatives à la télémédecine n'intègrent pas suffisamment les aspects de sécurités, même si les utilisateurs sont très préoccupés par leurs données. L'envoi de données via des canaux non protégés sur Internet les expose à un risque d'espionnage et d'altération par des attaquants malveillants. En particulier la modification des attributs d'un actionneur pour la santé peut être fatale dans certains cas (Simulateur cardiaque par exemple). Cependant, la sécurisation des échanges ne suffit pas pour faire face à toutes les attaques ciblant les données de l'utilisateur. Un utilisateur capable d'établir un canal sécurisé avec la ressource n'est pas toujours éligible pour effectuer toutes les opérations (modification, récupération, activation, etc.) sur la ressource de l'objet intelligent. De plus, un accès non autorisé aux ressources peut entraîner une perte de confidentialité, d'intégrité et de disponibilité des ressources. Par conséquent, l'authentification des utilisateurs et la restriction de l'accès à la ressource uniquement aux utilisateurs autorisés sont nécessaires. Dans nos propositions, tant pour l'architecture associant WebRTC et le WoT que pour l'architecture de middleware de prochaine génération, la sécurité est cruciale, en particulier lors du traitement de données critiques telles que relatives aux domaines médicaux de l'utilisateur. Pour cette raison, plusieurs solutions ont été envisagées afin de réduire la vulnérabilité du système. En commençant par la confidentialité et l'intégrité des données, l'authentification et enfin le contrôle d'accès.

4.3.4.1 Confidentialité et intégrité

Les contrôles d'accès restent pertinents pour la gestion de structures intelligentes impliquant plusieurs domaines, en l'occurrence celui de la télémédecine [177] [178] [179]. Cette évolution impose de nouvelles exigences et de nouveaux défis en matière de sécurité dans ces environnements dits intelligents [177]. Les modèles de contrôle d'accès tels que :

- ✓ Le contrôle d'accès obligatoire (MAC), Le modèle MAC a une politique de sécurité définie et gérée par une autorité et ne peut pas être modifiée par les utilisateurs. Ceci exclut les problèmes liés aux fuites d'informations observés dans le modèle DAC. Cela est principalement dû au fait que les utilisateurs ne sont pas autorisés à interférer avec la politique de contrôle d'accès [180]. Contrairement aux stratégies de contrôle d'accès discrétionnaires, les sujets d'une stratégie de contrôle d'accès obligatoire ne sont pas propriétaires des informations auxquelles ils ont accès. De plus, l'opération permettant la délégation de droits est contrôlée par les règles de la politique. Les sujets ne contrôlent plus les informations qu'ils manipulent. Le sujet n'a accès aux informations que s'il y est autorisé par le système [177]. Nous avons noté quelques limites qui sont : vulnérable aux canaux cachés, ne prend pas en compte le composant d'administration dans la gestion des rôles, ne prend pas en compte les problèmes de délégation et le niveau de confiance ;
- ✓ Le contrôle d'accès discrétionnaire (DAC) dont Les stratégies sont basées sur les concepts de sujets, d'objets et de droits d'accès. Les droits d'accès à chaque information sont manipulés par le propriétaire de l'information. Ce modèle de contrôle d'accès est flexible car un sujet avec des droits d'accès peut octroyer des droits d'accès à tout autre utilisateur. L'octroi ou la révocation de privilèges est régularisé par une politique administrative décentralisée [180]. Ce modèle a quelques limites : difficulté d'administration et limitation de l'accès aux objets en fonction de l'identité de l'utilisateur ;
- ✓ Le contrôle d'accès basé sur les rôles (RBAC) proposés jusqu'à présent sont statiques. Du fait qu'ils ne tiennent pas compte des règles spécifiques de l'organisation, ni de la gestion des obligations de recommandations, ni le coté dynamique du modèle d'accès, ni de la gestion des obligations recommandations, ni les règles spécifiques à l'organisation [181].

Afin d'améliorer les politiques de contrôle d'accès, les chercheurs ont travaillé sur des modèles de contrôle d'accès dynamiques tels que OrBAC, GeoRBAC (Contrôle d'accès basé sur les rôles géographiques), Contrôle d'accès basé sur les rôles contextuels (CRBAC), Multi-OrBAC, Poly OrBAC. Ces modèles représentent chacun une extension du RBAC, mais ne sont pas entièrement satisfaisants car ils ne permettent pas de gérer la délégation de rôles, en particulier dans un contexte de télémédecine qui nécessite la disponibilité de personnel traitant en temps réel. Dans notre proposition, nous avons implémenté le modèle DORBAC, qui est une

extension du modèle OrBAC, prenant en compte le problème de délégation de rôle et le problème d'administration pour l'attribution de licence et de rôle.

Les stratégies de contrôle d'accès discrétionnaire sont basées sur les concepts de sujets, d'objets et de droits d'accès. Les droits d'accès à chaque information sont manipulés par le propriétaire de l'information. Ce modèle de contrôle d'accès est flexible car un sujet avec des droits d'accès peut octroyer des droits d'accès à tout autre utilisateur. L'octroi ou la révocation de privilèges est régularisé par une politique administrative décentralisée [180]. Ce modèle a quelques limites : difficulté d'administration et limitation de l'accès aux objets en fonction de l'identité de l'utilisateur.

4.3.4.2 Contrôle d'accès de base de rôle (RBAC)

Le modèle de contrôle d'accès basé sur les rôles, ou RBAC, est considéré comme une approche alternative au contrôle d'accès obligatoire (MAC) et au contrôle d'accès discrétionnaire (DAC). Sa politique de sécurité ne s'applique pas directement aux utilisateurs [182] [183]. Le modèle RBAC est centré sur le rôle [184] [185] [186].

RBAC est un contrôle d'accès non discrétionnaire garantissant que l'accès n'est accordé qu'aux utilisateurs autorisés. C'est un modèle proposé par Ferraiolo et khun en 1992 [187], qui repose sur la création d'une relation entre le rôle d'un utilisateur dans un environnement donné et ce que ce rôle peut faire, ou en d'autres termes, les autorisations. Par exemple, dans de nombreuses organisations, les utilisateurs finaux ne sont pas propriétaires des informations, même s'ils disposent d'une autorisation d'accès. Pour ces organisations, la société ou l'agence est le propriétaire réel de l'information. Par conséquent, le contrôle de ces informations est souvent basé sur le travail de l'employé plutôt que sur la propriété des données. Les décisions de contrôle d'accès sont souvent déterminées par les rôles auxquels les utilisateurs sont affiliés dans le cadre de l'organisation. Cela inclut les différentes tâches, responsabilités et qualifications. Par exemple, les rôles d'un individu dans une infrastructure telle qu'un hôpital peuvent inclure : un médecin, une infirmière, un pharmacien, etc. Les stratégies RBAC sont basées sur les fonctions qu'un utilisateur est autorisé à exécuter dans l'organisation à laquelle il / elle est affiliée [188]. Un rôle peut être vu comme un ensemble de transactions qu'un utilisateur ou un ensemble d'utilisateurs peut effectuer dans une organisation. L'appartenance à un rôle est accordée et révoquée par un administrateur. Habituellement, les différentes transactions sont attachées aux différents rôles également par un administrateur. Ces transactions, par exemple, peuvent inclure la possibilité pour un médecin d'établir un diagnostic, de prescrire un

médicament et d'ajouter une entrée à un registre des traitements effectués sur un patient. Les limites sont :

- Aucune délégation de rôle [189] ;
- La protection de la vie privée n'est pas prise en compte ;
- N'exprime pas d'interdictions, de recommandations ou d'obligations.

4.3.4.3 Contrôle d'accès basé sur l'organisation (OrBAC)

Dans toute organisation, l'administrateur est responsable de la gestion de l'accès de chaque utilisateur à une ressource, en appliquant des règles de sécurité. Mais la gestion des droits d'accès devient complexe à mesure que le nombre d'utilisateurs, de ressources et d'activités augmente. Dans ce contexte, le modèle OrBAC résout ce problème en créant des entités abstraites (rôle, vue, activité) séparées en entités concrètes (sujet, objet, action). L'objectif de cette séparation est d'appliquer les règles de sécurité à des entités abstraites et à chacune de ces entités, une entité concrète est associée. OrBAC définit quatre types de règles de sécurité : autorisation, obligation, interdiction et recommandation [190].

Limitations : Aucune délégation ni préservation de la vie privée. La figure 4.4 ci-dessous montre le modèle OrBAC.

Figure 4.4: Structure du modèle OrBAC

Nous avons étudié les modèles de contrôle d'accès les plus célèbres. Chacun d'eux présente des avantages et des limites. Les modèles de contrôle d'accès statique, dont le plus avancé est RBAC, ont une limite importante en raison de son manque de dynamicité. Plusieurs modèles, tels que TrustBAC, TRBAC, ont été proposés dans le but de l'améliorer partiellement, mais aucun d'entre eux n'intègre, à notre connaissance, les paramètres relatifs à la délégation et à la protection de la vie privée. Nous avons également introduit des modèles de contrôle d'accès dynamiques, dont le plus avancé est OrBAC. Malheureusement, malgré son dynamisme

et sa capacité à gérer les autorisations, interdictions, obligations, recommandations, ce dernier ne tient pas compte de certains paramètres importants déjà présents dans les extensions de modèle RBAC et également comme RBAC, la délégation et la confidentialité des données. Les limites observées dans cette synthèse justifient notre choix de proposer un modèle dynamique prenant en compte la délégation et la confidentialité des données.

4.3.5 Scénario E-Health

De nombreux patients meurent dans les zones rurales en raison d'un manque de prise en charge par des médecins et aussi de manque de moyens. Parmi les maladies chroniques où le patient a besoin d'une surveillance continue on peut citer l'hypertension artérielle, diabète, maladie rénale, etc. Dans ces domaines, les structures de référence sont implantées dans les capitales régionales. Le coût de la prise en charge déjà élevé est majoré par les frais liés au déplacement et à l'hébergement, ce qui le rend insupportable pour la plupart des patients. La démocratisation de l'accès aux soins constitue ainsi un défi de taille lié au manque de spécialistes dans les régions. Pour la suite nous avons proposé des contributions pour surveiller les maladies chroniques, la pédiatrie et un modèle de contrôle d'accès afin de sécurisé les données des patients.

4.3.5.1 Contribution pour la surveillance des maladies chroniques

Pour satisfaire l'objectif d'accès universel à des soins de qualité et à moindre coût dans les zones rurales, une solution de monitoring portatif est proposée dans cette section. Un système basé essentiellement sur les capteurs biométriques et la Raspberry Pi est utilisé. Les infirmiers se trouvant dans les zones rurales l'utilisent pour collecter des informations biométriques recueillies par les capteurs telles que l'hypertension artérielles, le taux de sucre dans le sang, l'électrocardiogramme (ECG) et la température corporelle du patient. Si une des valeurs reçues des capteurs biométriques du patient devient critique, l'infirmier envoie un message à un médecin spécialiste qui se trouve dans sa région. Cela aide à prendre les mesures appropriées et d'anticiper sur les éventuels risques de maladies qui peuvent subvenir. Un tel système peut aider à réduire les factures d'hôpitaux découlant de l'admission du patient à l'hôpital et l'accès à de soins de qualités tout en restant dans sa localité.

- **Architecture de la solution**

L'architecture de notre solution s'articule autour de 5 entités qui sont :

- ✓ Les capteurs ;

- ✓ La raspberry Pi et le kit e-Heath ;
- ✓ La base de données ;
- ✓ Application.

La figure suivante nous montre une description de la solution proposée.

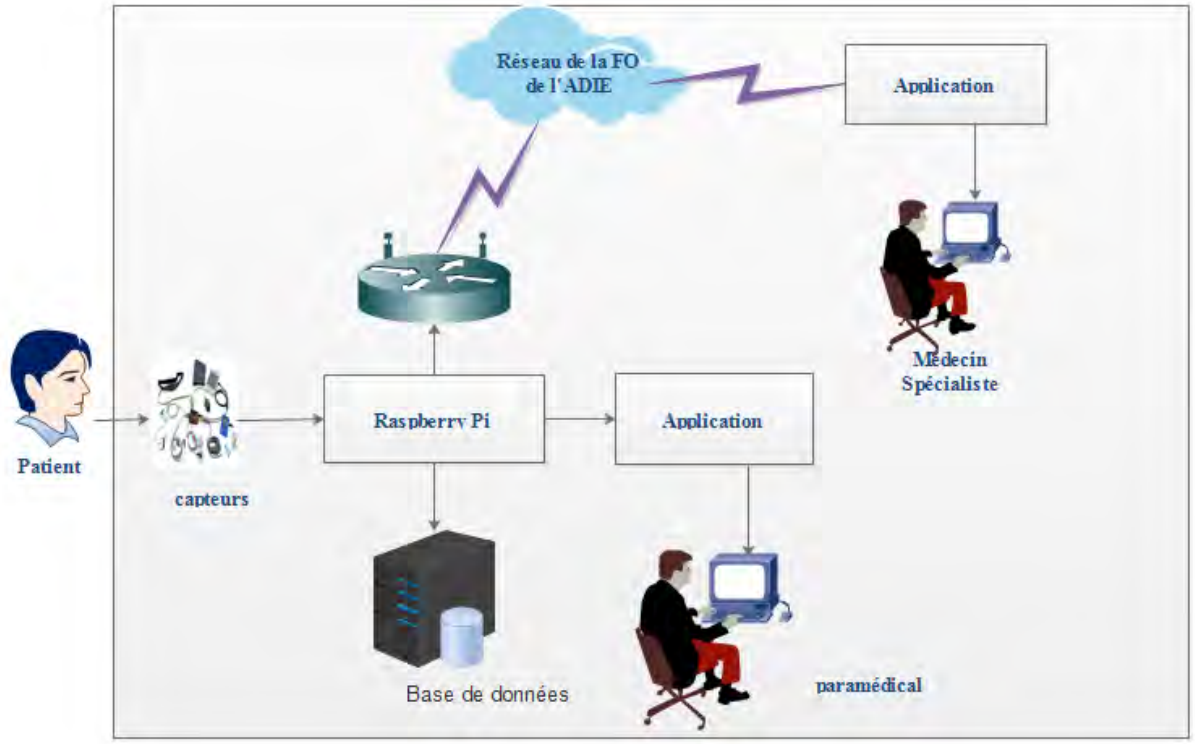

Figure 4.5: Architecture de monitoring des maladies chroniques

Le système d'accès universel de soins dans les zones rurales est basé sur les formes d'acquisition des signaux vitaux des patients. Pour cela différents capteurs sont utilisés. Sur le kit e-Heath V2.0 [191], qui est une carte permettant aux utilisateurs d'Arduino et Raspberry Pi d'effectuer une surveillance médicale.

La raspberry Pi est utilisé pour recevoir et analyser les signaux acquis par les différents capteurs et les envoie sur les personnels de santé. Une carte Raspberry est en fait un micro-ordinateur qui n'a rien à envier à ses grands frères. En effet, à l'inverse des cartes dites « programmables » telles que les cartes Arduino, les Raspberry possèdent tous les composants d'un ordinateur classique : sorties audio et vidéo, ports USB, lecteur de carte SD, connexion internet, etc. Ces cartes ont pour avantages d'être très compactes de pouvoir être utilisées dans de nombreux domaines (ordinateur de poche, serveur de petite taille, domotique…), tout en étant très économique. Le choix du système d'exploitation de la carte se fait parmi un large choix de distributions Linux spécialement développées pour les Raspberry [192].

Une Base de données Relationnelle est mise sur place pour stocker les données biométriques des patients, des médecins spécialistes et des médecins généralistes. Le traitement prescrit et les conseils du médecin généraliste ou spécialiste sont également disponible dans la base de données. Les infirmiers et les médecins peuvent accéder à l'information chaque fois que nécessaires.

L'application basé sur le web utilisant PHP et HTML est responsable de l'architecture client-serveur permettant une communication entre les paramédicaux et les médecins spécialistes [193] [194].

Clients : Le côté client permet l'interaction entre infirmiers et des médecins. Les infirmiers sont les acteurs de l'acquisition des données biométriques du corps des patients en utilisant différents capteurs. Ces signaux sont transmis sur le réseau dédié par exemple celui de la fibre optique de l'Agence Informatique de l'Etat sénégalais (l'ADIE), via un support de communication à des médecins spécialistes ou généralistes de la région polarisant le poste de santé. Une application web est utilisée pour visualiser les signaux biométriques comme l'HTA, le Taux de sucre dans le sang du patient en temps réel [195].

Lorsque l'infirmier remplit les informations relatives au patient, un code lui est envoyé pour l'identifier de manière unique dans le réseau. L'Id lui permettra de se connecter sur son compte plus tard. Les données sont non seulement surveillées mais également comparées aux valeurs normales déjà définies. Le médecin spécialiste accède à toutes les informations avec l'image des signaux capté par les capteurs posés sur le patient. Par conséquent le médecin une fois qu'il a les informations du patient, il les traite et prodigue des traitements et des conseils à l'infirmier et au patient.

Coté serveurs : Il est utilisé pour rendre disponible et stocker les données du patient et du médecin. Si le médecin, l'infirmier et le patient veulent se communiquer le système est capable de leurs fournir une communication par WebRTC. Pour cela il faut connecter une caméra sur la raspberry Pi et le médecin spécialiste doit disposer d'un appareil supportant le WebRTC.

- **Description de la solution**

Les utilisateurs incontournables du système sont les infirmiers, les médecins spécialistes et les patients. La figure 4.6 nous montre le scénario du système propose.

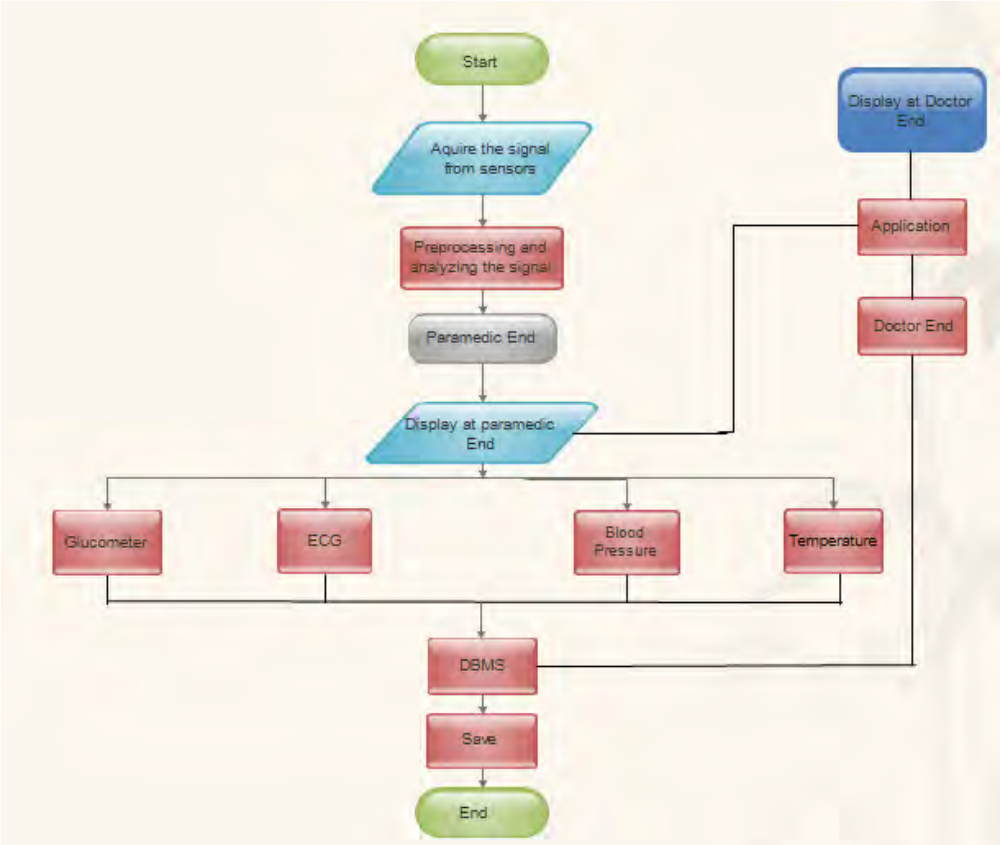

Figure 4.6: organigramme du model propose

Le paramédical remplit les renseignements sur les patients tel que : le Nom, Prénom, l'adresse, l'âge, le sexe, N° de téléphone domaine de la maladie. Il met les capteurs sur le patient afin de recueillir ses constantes biométriques. Une fois ces informations collectées, l'ensemble est transmis dans la base de données et accessible partout dans le réseau prévu à cet effet (exemple celui de l'ADIE). Après cette étape, le paramédical sera en mesure de mettre en relation le patient à un médecin spécialiste.

Ce système proposé est conçu pour fonctionner entre le médecin et les paramédicaux. La tâche principale sera de surveiller et de recueillir l'état de santé du patient. Le médecin spécialiste peut prescrire au patient un traitement et lui donner aussi des conseils par vidéoconférence ou par chat. L'infirmier télécharge le fichier de traitement suivant le format voulu PDF ou autre.

L'évaluation des données est une question importante dans les systèmes de surveillance des patients. Les données sont non seulement surveillées mais également comparées aux valeurs normales déjà définies.

Si les données sont hors de la gamme normale de valeurs prédéfinie, l'infirmier envoie un message d'alerte au médecin spécialiste de la maladie. Ce dernier peut recevoir un message sur sa boite email et en SMS via un module GSM incorporé dans le système.

4.3.5.2 Contribution à la Télépédiatrie

L'objectif de cette proposition s'accentue sur la télé pédiatrie afin de contribuer à la baisse du taux de mortalité infanto-juvénile en milieu rural qui reste élevé aujourd'hui. Pour ce faire, nous couplons l'intelligence du WoT à la puissance du WebRTC. Cette plateforme, basée sur le serveur multimédia WebRTC Kurento et le Web of Things (WoT), permet aux médecins pédiatres et aux patients, enfant accompagné par sa mère de réaliser des consultations à distance. Kurento Media Serveur (KMS) permet de créer des applications de traitement de média basées sur le concept de pipelines. Le Web of Things (WoT), considéré comme un sous-ensemble de l'Internet des objets (IoT), se concentre sur les normes et les frameworks logiciels tels que REST, HTTP et URI pour créer des applications et des services qui combinent et interagissent avec une variété de périphériques réseau.

❖ **Description de la solution**

Pour exploiter les potentiels de WebRTC et IMS, nous proposons leur integration en vue de permettre à l'un de l'autre. Il consiste à ajouter de nouvelles fonctionnalités pour contrôler, gérer et envoyer des données à partir des objets connectés disponibles dans l'environnement du point de terminaison WebRTC, Kamailio et IMS. L'objectif est au cours d'une session multimédia WebRTC de pouvoir accéder aux capteurs et d'envoyer des données de l'autre côté de la communication en temps réel et aussi de pouvoir bénéficier des appels simples avec des utilisateurs IMS-IMS, IMS-KMS et KMS-KMS. Le Coeur de réseau Kamailio-IMS et le serveur multimedia Kurento peuvent être déployés sur un serveur au sein d'un réseau pouvant supporté la solution. Il y a donc deux cas de figure :

✓ **Clients connectés au réseau dédié sans nécessité d'internet**.

Dans ce cas, la mère se déplace dans un centre hospitalier équipé à cet effet. Au sein de cette structure se trouvent des ordinateurs, des tablettes ou des téléphones portables connectés au réseau dédié (par exemple celui de l'ADIE pour le Sénégal).

En zone urbaine aussi se trouve le même dispositif prêt à recevoir un appel de la mère pour une consultation de pédiatrie avec un pédiatre. Une fois les formalités administratives remplies, l'appel est lancé sous la supervision d'un(e) infirmier/infirmière et la communication

est établie entre le médecin et la mère. La figure 4.7 montre le mécanisme de communication entre la mère et le spécialiste (Pédiatre) grâce au réseau dédié sans connexion internet.

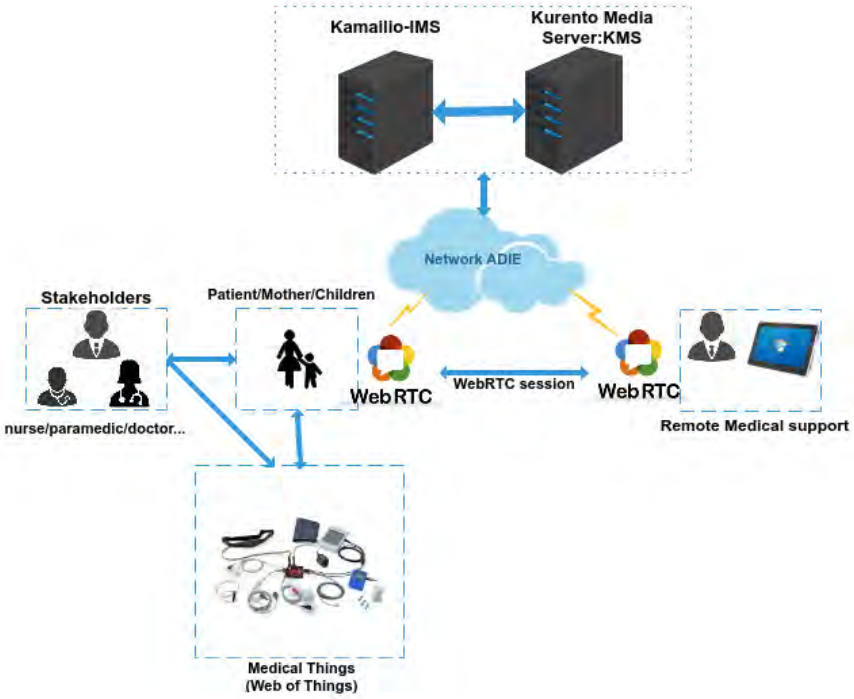

Figure 4.7: Architecture K-2I-E-health sans connexion internet

✓ **Clients externes au réseau avec connexion internet**

La seule différence avec le cas précédent est que la mère ne se déplace pas dans un centre hospitalier équipé techniquement. A l'aide de son Smartphone, sa tablette ou son ordinateur connecté à l'internet, elle peut se connecter au système et émettre ou recevoir un appel respectivement en direction ou venant du spécialiste en pédiatrie après que les formalités administratives soient remplies. L'illustration de la communication est donnée par la figure 4.8.

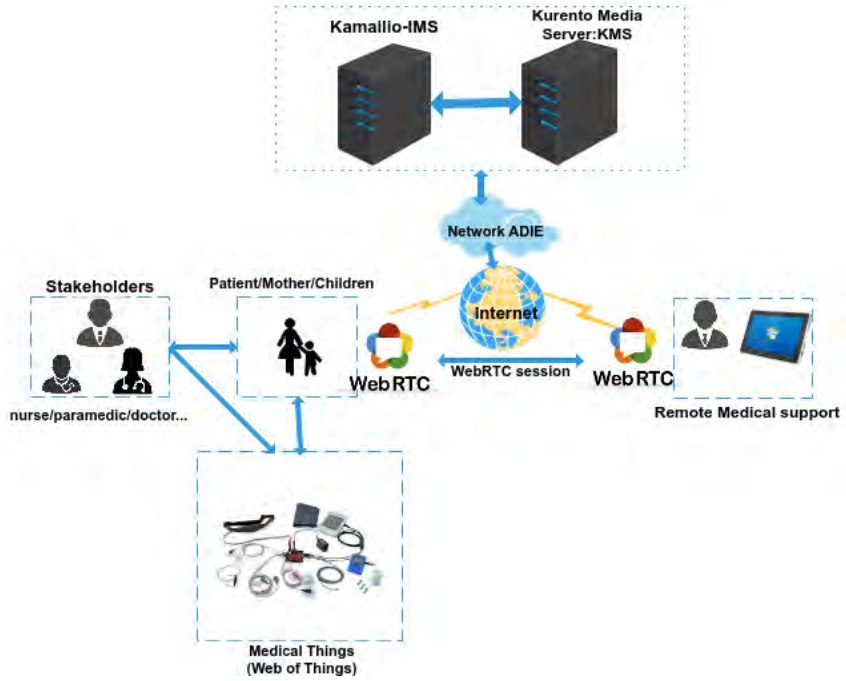

Figure 4.8: Architecture K-2I-E-health avec connexion internet

❖ **Implémentation**

L'architecture proposée est composée de trois entités distinctes : WoT, API et Application Web.

La première partie est la partie WoT. Chaque point de terminaison est considéré comme une passerelle vers son ensemble d'objets intelligents. En outre, chaque utilisateur a un contrôle sur ces objets. Le nœud d'agrégation NODEMCU ESP8266 (Raspberry) n'est pas responsable de la lecture des capteurs. Il fournit simplement une passerelle entre l'utilisateur et le réseau de capteurs, puis effectue des analyses de données. Le nœud de capteurs est le niveau le plus bas d'un réseau de capteurs. Il est chargé de recueillir des informations auprès des capteurs, d'effectuer des actions de l'utilisateur et d'utiliser des mécanismes de communication pour envoyer des données au nœud d'agrégation (voir figure 4.9).

Figure 4.9: Architecture WoT

La passerelle ESP8266 peut alors communiquer avec les capteurs en utilisant l'un des protocoles de communication bien connus (Lora, Zigbee, Bluetooth, WIFI, etc.).

Dans le cas des scénarii d'e-santé, tout ce dont nous avons besoin est des capteurs médicaux portables. Ils peuvent communiquer via n'importe quel protocole, puisque le WoT résume toute la complexité de la connectivité des objets.

En utilisant l'architecture actuelle, une mise en œuvre de l'examen clinique à distance est possible, où un médecin communique avec un patient en utilisant Kurento Media Server. Le médecin spécialiste ou généraliste a accès à un ensemble de capteurs. Ensuite, il peut envoyer les informations recueillies par ces capteurs au médecin spécialiste en temps réel en utilisant la plateforme K-2I-E-health. Enfin, ces données pourront être analysées et commentées par les acteurs.

La deuxième partie est l'API. Nous avons développé une API REST capable de récupérer les informations collectées par un appareil médical connecté et de les stocker dans une base de données MongoDB. MongoDB appartient à la famille NoSQL Document-store, développée en C++. Il est basé sur le concept de pair clé-valeur. Le document est lu ou écrit en utilisant la clé. MongoDB prend en charge les requêtes dynamiques sur les documents. Comme il s'agit d'une base de données orientée document, les données sont stockées sous la forme de JSON, style BSON [196].

La troisième entité de la plateforme est l'application Web. Pour mettre en place l'application web, nous utilisons les technologies NodeJs et Kurento Media Server. Cette plateforme permet aux médecins et aux patients de s'enregistrer puis de s'authentifier afin d'accéder aux fonctionnalités de Kurento Media Server. Une fois connectés, le médecin spécialiste (pédiatre) peut visualiser les données des capteurs et les flux médias du patient. La figure 4.10 montre le principe d'authentification sur la plateforme K-2I-E-health

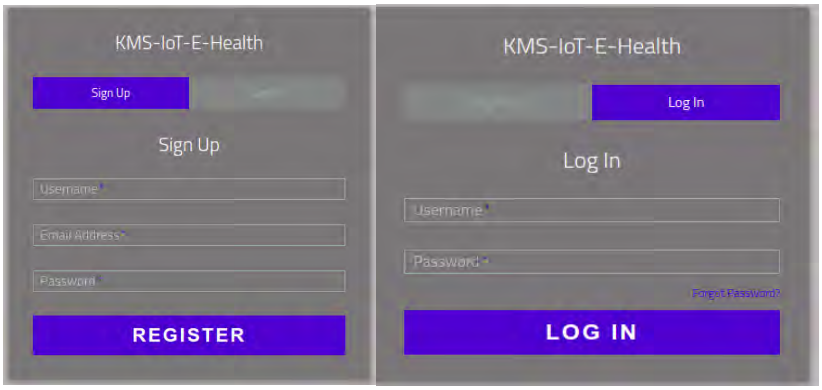

Figure 4.10: Authentication et login sur la plateforme K-2I-E-health

L'application web permet également de recueillir des informations de la base de données et de les afficher. Les utilisateurs connectés peuvent alors visualiser les données des capteurs.

❖ **Mécanisme de communication entre client et médecin sur la plateforme**

Pour établir une connexion WebRTC entre les clients (Patient et Médecin) et le serveur multimédia Kurento des candidats SDP (Session Description Protocol) et ICE (Interactive Connectivity Establishment) doivent être échangés. Plus précisément, la négociation SDP connecte WebRtcPeer dans le navigateur avec WebRtcEndpoint sur le serveur d'application. Pour effectuer un appel, chaque utilisateur doit être enregistré dans le système grâce à une classe UserRegistry pour stocker et localiser le patient et médecin.

Afin de contrôler les fonctionnalités multimédia fournies par le serveur Kurento, nous avons besoin d'une instance de KurentoClient dans le serveur d'applications. Pour créer cette instance, nous avons spécifier l'emplacement de Kurento Media Server dans la bibliothèque client. Une fois le client Kurento instancié, le patient et le médecin sont prêt à communiquer avec Kurento Media Server. Les « Media Pipelines » et les « Media Elements » sont créer et connecter. Dans ce cas de figure, nous avons besoin de deux points WebRtcEndpoint, à savoir le patient et médecin. Cette logique de média est implémentée dans la classe CallMediaPipeline.

A partir de Kurento Media Server 6.0, la négociation WebRTC est réalisée en échangeant des candidats ICE entre les homologues WebRTC. Pour implémenter ce protocole, WebRtcEndpoint reçoit les candidats du client dans la fonction OnIceCandidate. Ces candidats sont stockés dans une file d'attente lorsque le WebRtcEndpoint n'est pas encore disponible. Ensuite, ces candidats sont ajoutés à l'élément multimédia en appelant la méthode addIceCandidate. Une fois la session établie, le médecin spécialiste (pédiatre) peut visualiser les données des capteurs et les flux médias du patient sur la plateforme.

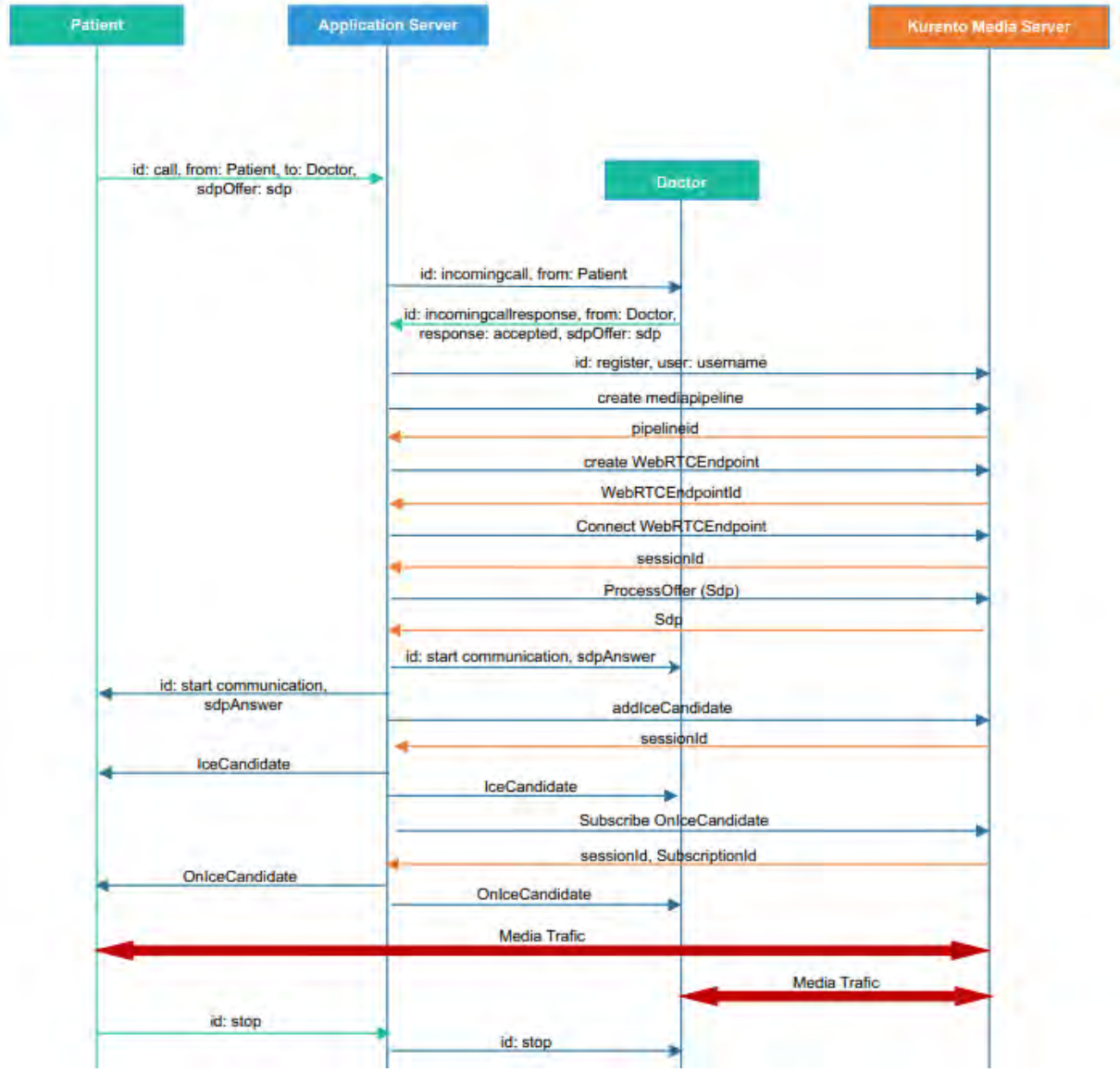

Figure 4.11: Diagramme de communication entre Patient et médecin sur la plateforme K-2I-E-health

4.3.6 Modèle de sécurité basé sur les controles d'accès (DORBAC)

Dans cette contribution, nous avons proposé le modèle DORBAC, qui est une extension du modèle OrBAC. Notre modèle permet de prendre en compte la délégation de rôles tout en assurant la protection de la vie privée du patient. Ainsi, le modèle proposé ne permet la délégation qu'entre médecins. L'application de notre modèle ne permet pas la saisie des données du patient par un agent de santé ou un assistant. Le risque d'erreur de saisie est ainsi éliminé et la confidentialité du patient préservée. Une fois le patient connecté aux capteurs, ses données sont analysées par le réseau de capteurs à l'aide de la Passerelle ESP8266, collectée par le nœud du capteur avant d'être stockée dans la base de données. Le patient ou le médecin disposant des

autorisations requises peut visualiser les données stockées via l'interface de l'application. Le médecin peut suivre le patient et prendre une décision en fonction des informations reçues des capteurs. Une séance de vidéoconférence est alors possible entre le patient et le médecin.

4.3.6.1 Description de la logique non monotone T-JClassic$\delta\varepsilon$

La logique non monotone T-JClassic$\delta\varepsilon$ a été développée pour permettre une meilleure gestion de l'aspect temporel dans divers domaines tels que le raisonnement sur les actions et les plans, l'amélioration de la compréhension des langages naturels et l'amélioration du contrôle d'accès. **T-JClassic$\delta\varepsilon$** permet de représenter des concepts temporels tout en ayant une connaissance par défaut. Cela diffère des logiques de description temporelles existantes où des composants temporels sont ajoutés aux logiques de description classiques. **T-JClassic$\delta\varepsilon$** est composé d'un ensemble de concepts atomiques **P** et de rôles atomiques **R**, les deux constantes $\top$ (Top) et $\bot$ (Bottom) qui représentent respectivement le concept universel et le concept bas-tom, un ensemble d'individus appelé 'classique des individus ', les concepts C et D, les connectives unaires $\boldsymbol{\delta}$ (défaut) et **ε** (exception), la conjonction binaire **π**, le quantificateur permettant une quantification universelle des valeurs de rôle et le qualificatif temporel @ représentant l'intervalle' **X** 'à auquel un concept C s'applique, **u** est un nombre réel, n'est un entier, "**Ii**" sont des "individus classiques" [179].

4.3.6.2 Description du modèle proposé

Le modèle de contrôle d'accès basé sur la délégation et l'organisation (DORBAC) est une extension du modèle OrBAC. L'élément central de ce modèle est la délégation, tout en tenant compte de la confidentialité. Nous décrivons notre modèle dans un environnement de santé en ligne dans lequel plusieurs infirmières et médecins sont impliqués. Alors que l'infirmière joue un rôle clé dans la manipulation des données des patients, le rôle de ces dernières se limite au niveau matériel. Il sera ensuite responsable de la connexion des capteurs aux patients et ainsi les informations collectées seront stockées directement dans une base de données. L'axiome1 définit le médecin par défaut en tant que membre du personnel comme suit :

$$\text{Doctor} \equiv \text{Staff\_Member} \prod \text{Attribute\_Member} \prod \text{Licence\_assigment} \prod \text{Role\_Assigment} \prod \delta\text{Permission} \quad (1)$$

Le concept Doctor (médecin) est un membre du personnel (Staff_Member) qui dispose d'attributs(Attribute_Member), d'un rôle(Role_Assigment) qui lui a été attribué et d'une licence (Licence_assigment) qui lui permet de déléguer ses permissions.

La définition de l'utilisateur Doctor lui donne le droit d'accès aux services de l'environnement des objets connectés. Chaque médecin reçoit une licence lui permettant de déléguer son rôle à un autre médecin possédant les mêmes attributs et / ou attributs supplémentaires. L'attribution de rôle peut être considérée comme la première étape de l'autorisation. La cession de licence est considérée comme la deuxième étape. Cela donne le droit à un utilisateur, de déléguer son rôle à son collègue, qui possède les mêmes attributs que lui. La définition d'un rôle et d'une autorisation se traduit par les axiomes suivants :

Rôle: étant donné l'univers de toutes les autorisations, le rôle R est l'ensemble d'autorisations finies. En d'autres termes

$$R = \Sigma Pi \,/\, Pi \in UP \qquad (2)$$

Permission : Etant donné $\mathbf{U_{OIoT}}$ l'univers de tous les objets de l'IoT, $\mathbf{U_s}$ l'univers des services offerts par les objets connectés et $\mathbf{U_{OPS}}$ l'univers de toutes les opérations autorisées pour un sujet, une permission **P** est représentée par le triplet ($\mathbf{O_i, S_i, OPS_i}$) où $Oi \in U_{OIoT}$, $S_i \in U_s$ et $OPS_i \in U_{OPS}$.

$$P = \Sigma O_i + \Sigma S_i + \Sigma OPS_i \qquad (3)$$

$$\delta\text{Permission} = \text{ObjectConntedP.permission} \prod \text{ServiceP.permission} \prod \text{OperationP.permission} \qquad (4)$$

- **Attribution de rôle et de licence**

Assignation de rôle pour le médecin :

$$\delta\text{Role\_Assigment} \sqsubseteq \text{OrgR.Assignee} \prod \text{AssigneeR.assigment} \prod \text{RoleR.assigment} \prod \delta\text{PrivilegesR.Service} \prod \delta\text{PrivilegesR.ObjectConnected} \qquad (5)$$

Licence du docteur

$$\delta\text{Licence\_Assigment} \sqsubseteq \text{OrgL} \prod \text{AssigneeL.assigment} \prod \text{LicenceL.assigment} \prod \delta\text{PrivilegesL.Action} \prod \text{CibleL.Objet} \prod \text{ContextL} \qquad (6)$$

- **Délégation de rôle**

Dans cette section, nous considérons la délégation totale des rôles dans laquelle les médecins ayant les mêmes attributs peuvent se déléguer des rôles. Les attributs représentent l'ensemble des caractéristiques permettant de déterminer un sujet, un service ou un objet. Un médecin peut également déléguer son rôle à un autre médecin ayant plus d'attributs que lui. La délégation des rôles est représentée par l'axiome suivant :

Empower⊑ UserRD.Role_Delegation∏AttributeRD.Role_Delegation∏AssignmentRD.Role∏
ServiceRD.Service ∏ Object_ConnectedRD.Object ∏ AssigneeRD.Grantor ∏ Working_HourRD.hour (7)

La révocation de délégation peut être représentée comme suit:

δPermission⊑ UseL.License_Delegation∏AssigneeL_Assignee∏AttributeRD.Role_Delegation∏
DurationEndL.Licence_Delegation ∏ PermisionD.GD_Revoke (8)

- **Révocation de rôle**

La protection de la vie privée est une autre question importante [197] à prendre en compte dans des domaines tels que la gestion de crise. Dans le cadre de la gestion des données collectées via des objets connectés (capteurs), la protection de la vie privée dans le contrôle d'accès prend en compte deux dimensions, à savoir la vie privée de l'objet connecté et la vie privée du sujet (patient).

Préserver la confidentialité de l'objet connecté est un problème de confiance proche. Ainsi, un enregistrement stocké dans une base de données via des capteurs est protégé par un utilisateur si l'utilisateur demande l'accès aux informations à une fin autre que celle qui lui est associée. La protection de la vie privée de l'objet indique que l'accès d'un sujet affectera les attributs qui lui sont attribués. La vie privée du patient est préservée en ce que les informations sont reçues par le capteur et stockées directement dans la base de données sans l'intervention d'un agent de saisie de données.

- **Fonction d'affectation (assignation) d'objectifs**

Purp_assign (subject.ATTR, O_{IOT}.ATTR, Ops.ATTR, service.ATTR) = purp_attr (subject.ATTR) ⊆purp_attr (service.ATTR) ⊆ purp_attr (O_{IOT}.ATTR) ⊆ purp_attr (service.ATTR) ∈ {0, 1} **(9)**

Purp_attr est une fonction qui retourne l'attribut de l'ensemble des objectifs d'un sujet, d'un objet connecté, d'un service ou d'une opération.

Dans cette contribution, nous avons proposé le modèle DORBAC, qui est une extension du modèle OrBAC. Notre modèle permet de prendre en compte la délégation de rôles tout en assurant la protection de la vie privée du patient. Ainsi, le modèle proposé ne permet la délégation qu'entre médecins. L'application de notre modèle ne permet pas la saisie des données du patient par un agent de santé ou un assistant. Le risque d'erreur de saisie est ainsi éliminé et la confidentialité du patient préservée. Une fois le patient connecté aux capteurs, ses données sont analysées par le réseau de capteurs à l'aide de la Passerelle ESP8266, collectée par le nœud du capteur avant d'être stockée dans la base de données. Le patient ou le médecin disposant des

autorisations requises peut visualiser les données stockées via l'interface de l'application. Le médecin peut suivre le patient et prendre une décision en fonction des informations reçues des capteurs. Une séance de vidéoconférence est alors possible entre le patient et le médecin.

4.3.7 Contribution à amélioration de la base de présence des serveurs VoIP

La place du protocole SIP est importante dans un système de communication en temps réel plus précisément dans les middleware IoT intégrant IMS. Or dans notre architecture les clients ont la possibilité d'utiliser trois types de terminaux pour communiquer :

- WebRTC-WebRTC ;
- WebRTC- IMS ;
- IMS-IMS.

Donc, une importance capitale est accordée à l'étude de la base de présence des serveurs VoIP utilisant le protocole SIP. L'architecture de la configuration de base d'un serveur VoIP utilisant le protocole SIP est représentée sur la figure 4.12 ci-dessous

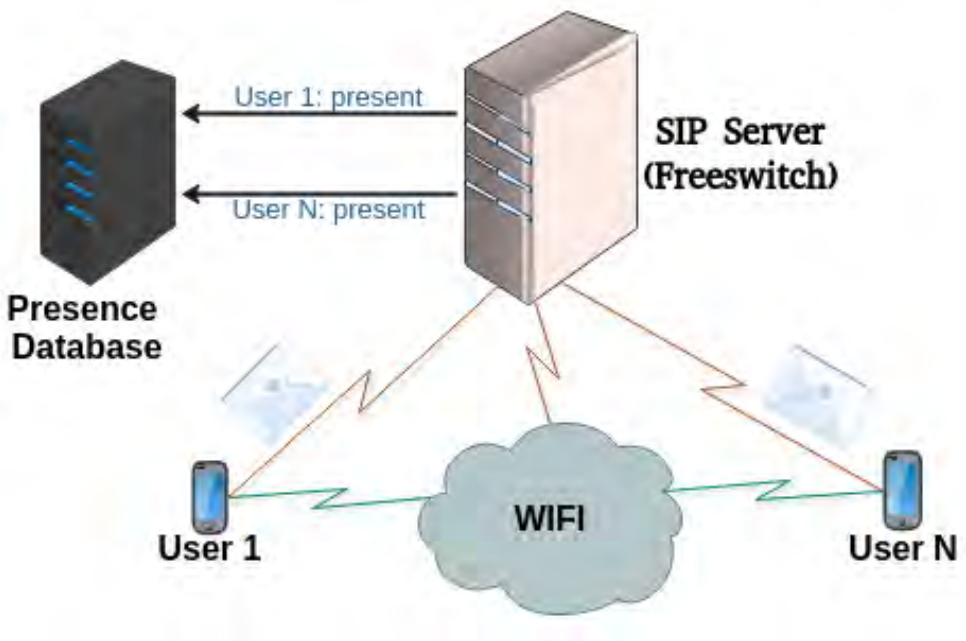

Figure 4.12: Architecture initial d'un serveur SIP

Dans cette architecture le User 1 et le User N sont tous connectés au serveur de téléphonie SIP qui dispose une base de présence où il stocke tous les utilisateurs connectés. Une foi un utilisateur se connecte, le serveur détecte sa connexion et l'enregistre au niveau de la base de présence donc il dispose des informations de sa présence, c'est-à-dire des événements

REGISTER. Une foi le User 1 connecté envoi un message au User N, le serveur consulte sa base de présence pour avoir les informations sur le User N. S'il voit que celui-ci est présent au niveau de la base de présence, il lui retransmet le message qui lui est destiné. Sinon si le User N est absent au niveau de la base de présence, le message qui lui était destiné est perdu. Les figures suivantes nous montrent des illustrations des failles notées sur les serveurs de présences des utilisateurs dans les PBX VoIP.

- **Envoi de message entre user : tous les deux connectés (user 1000 et user 1001)**

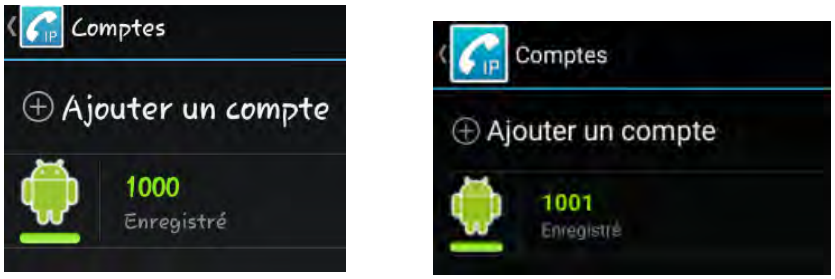

Figure 4.13: Enregistrement des user 1000 et 1001

Le user 1000 envoie un message à destination du user 1001 connecté.

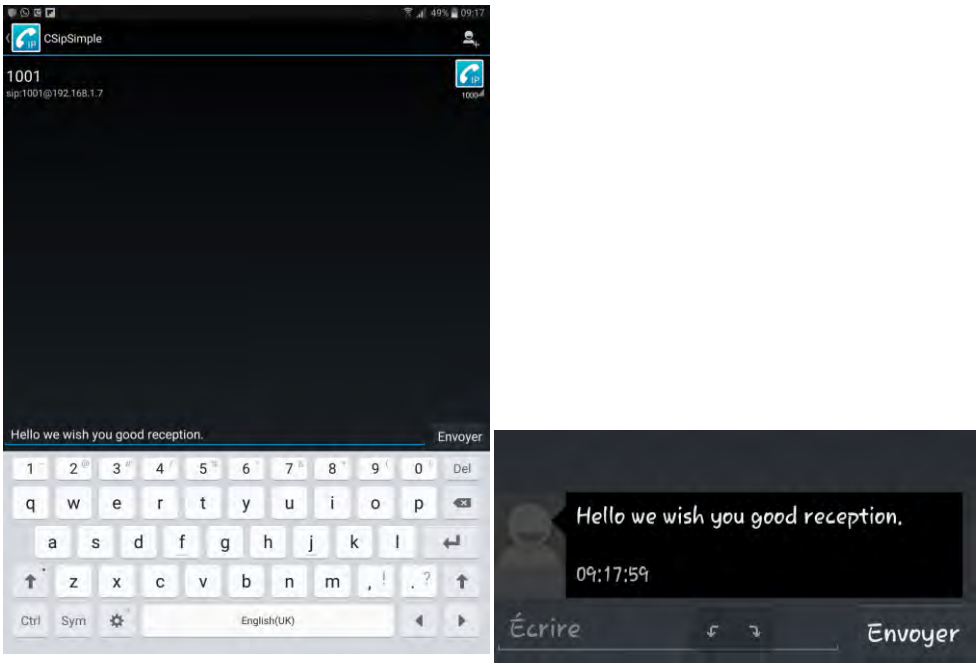

Figure 4.14: Envoi et réception de message du user 1000 à destination du user 1001

Pour cette configuration initiale du serveur SIP, si tous les utilisateurs sont connectés au serveur SIP, les messages envoyés arrivent toujours à leur destinataire comme nous pouvons le constater pour le message envoyé par le user 1000 à destination du user 1001. Cet exemple est

le cas où les deux utilisateurs sont connectés, nous allons voir maintenant le cas où l'un des utilisateurs n'est pas connecté.

- **Envoi de message entre user: l'un connecté (1000) et l'autre déconnecté (1001)**

Le user 1000 envoi un message à destination du user 1001 non connecté.

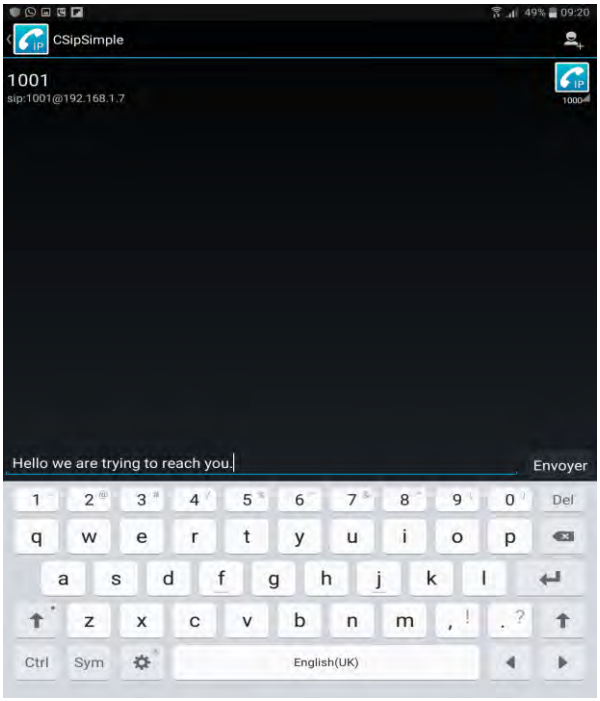

Figure 4.15: Envoi de message du user 1000 à destination du user 1001 non connecté

```
2018-06-13 09:21:50.876536 [ERR] sofia_presence.c:272 Chat proto [sip]
from [<sip:1000@192.168.1.7>;tag=vrsoUgywr48sO7etKy9cCi1.eul3r9OZ]
to [1001@192.168.1.7]
Hello we are trying to reach you.
Nobody to send to: Profile internal
```

Figure 4.16: Informations du côté du serveur

Après l'envoi de ce message, nous allons consulter notre serveur et ceci grâce à sa console, nous observons les messages d'erreur.

La figure 4.16, montre les messages au niveau des logs de notre serveur et que le message envoyé au user 1001 n'est pas arrivé à destination. On note que ce message envoyé en absence de connexion du User N n'est pas arrivé à destination. On constate qu'un autre problème, si le user désactive son voyant WIFI, le serveur n'a pas des informations de déconnexion pour pouvoir déconnecter correctement le user de la base de présence. Du coût, il continu à le voir comme celui-ci était toujours connecté au serveur. La figure ci-dessous met en œuvre le phénomène énuméré.

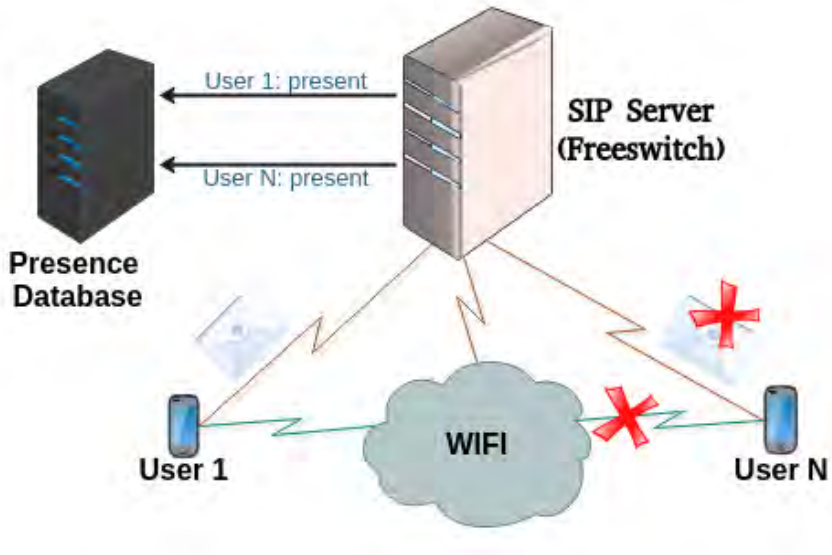

Figure 4.17: Architecture sur le problème de connexion noté

Le User N décide de désactiver son voyant WIFI et comme nous le constatons le serveur continue de l'enregistrer comme présent au niveau de sa base de présence. Le message envoyé par le User 1 au User N, n'est pas arrivé à destination. Cela est dû tout simplement du fait que le serveur a toujours les informations d'enregistrement du User N dans sa base de présence, donc il essaye de l'envoyer le message alors qu'il n'est pas en réalité connecté. Du coût ce message ne serai pas reçu par le User N.

Les figures suivantes présentent les différentes étapes qui ont conduite à cette étude. Dans cette figure tous les deux utilisateurs sont connectés au serveur de téléphonie. Ils sont présents au niveau de la base de présence.

	call id	sip user	sip host	presence hosts	contact	st
1	QWN-Q4PPA755	1000	192.168.1.7	192.168.1.7,192	"" <sip:1000@1	Re
2	c7gHjeB2EQsXO	1001	192.168.1.7	192.168.1.7,192	"" <sip:1001@1	Re

Figure 4.18: La base de présence du serveur de téléphonie

L'expérience mis en œuvre est que si le user 1001 décide de désactiver uniquement sont voyant WIFI. Si nous actualisons notre base de présence, nous continuons à avoir sa présence au niveau de celle-ci. Le user 1000 va lui envoyer un message. On constate au niveau des logs de notre serveur SIP que le message envoyé n'arrive pas à destination.

La figure suivante montre le message envoyé par le user 1000 au user 1001.

```
2018-06-13 09:42:32.355696 [INFO] switch_cpp.cpp:1365 *******   Evenement Message    *********
2018-06-13 09:42:32.355696 [INFO] switch_cpp.cpp:1365 ipv4 : 192.168.1.7
2018-06-13 09:42:32.355696 [INFO] switch_cpp.cpp:1365 sender : 1000
2018-06-13 09:42:32.355696 [INFO] switch_cpp.cpp:1365 recipient : 1001
2018-06-13 09:42:32.355696 [INFO] switch_cpp.cpp:1365 message : Hello you're there?
```

Figure 4.19: Message envoyé vers 1001 qui désactive son WIFI

Nous constatons que le message arrive au niveau du serveur mais le user ne va jamais le recevoir. Pour pallier à cela, nous avons proposé une solution qui est décrite en détaille avec l'architecture qui a été mise en place au niveau de la section suivante. Cela nous amène à faire une étude approfondie du SIP pour apporter une solution à ce genre de problème.

4.3.7.1 Implémentation

Pour implémenter cette solution, nous avons installé un serveur SIP freeswitch que nous avons exploité en envoyant des requêtes et grâce à son module ESL (*Event Socket* Library), nous parvenons à écouter sur les événements de type MESSAGES. Cette écoute des événements nous permet de déceler les différentes failles notées dans la section précédente. Pour résoudre les failles, nous avons développé un script en python et adapter aux serveurs SIP comme freeswitch et astérisk, on a mis en place aussi une base de données MySQL qui va servir pour stocker les messages dont les propriétaires ne sont pas enregistrés au niveau du serveur SIP. Dans ce script, nous écoutons sur les événements MESSAGES de freeswitch puisse que les failles que nous avons notées concerne des problèmes de réception de message en cas de mauvaise connexion. Ce script écoute en boucle infini sur ces événements, il nous permet de récupérer tout événement qui arrive sur notre serveur de téléphonie afin de pouvoir l'exploiter, soit on l'envoi au destinataire soit c'est à stocker dans une base de données MySQL. Cette base de données nous permet aussi d'avoir les traces des différents messages échangés. Dès que le destinataire qui était absent se reconnecte, notre script reçoit l'événement message de reconnexion, il vérifie dans la base de données si l'utilisateur en question n'a pas de messages stocker en absence. Si c'est le cas alors le script récupère le(s) message(s) de la base de données et le reformate pour l'envoyer à son destinataire. Si ce n'est pas le cas, le script notifie dans la console du serveur SIP que l'utilisateur n'a pas de message en absence. L'architecture proposée est donnée à la figure suivante.

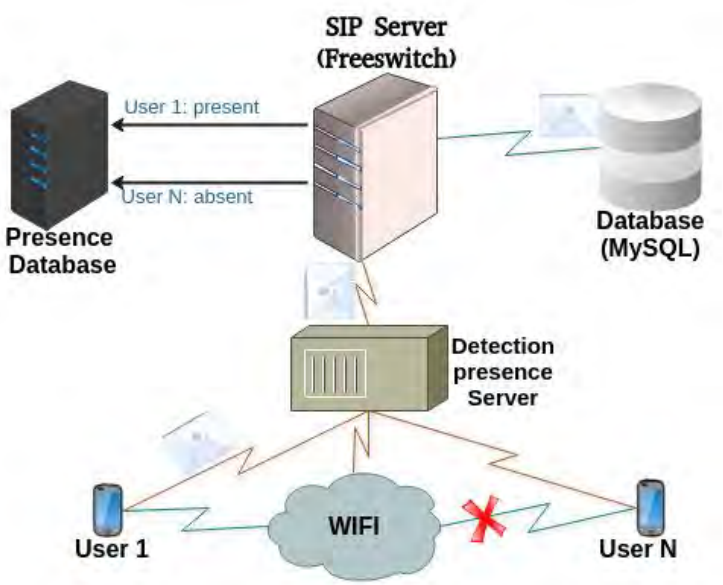

Figure 4.20: Architecture de la solution proposée

4.3.7.2 Résultats et discussion

Suite au problème décelé précédemment, nous allons proposer une solution qui nous permettra de pallier à cela. L'architecture ci-dessous montre comment la solution a été mis en œuvre et une description de cette solution sera apporté. Dans cette architecture, on constate la présence d'un autre serveur qu'on appelle ici « Serveur de détection de présence ». Ce serveur fonctionne comme suit :

Il envoi tous les cinq secondes un **Ping** aux utilisateurs connectés au serveur SIP.

- Si le User N, qui était jusque-là connecté au serveur, décide de désactive son voyant WIFI, automatiquement le serveur de détection de présence en lui envoyant un Ping obtient comme réponse UNREGISTERED.
- Il contacte alors le serveur SIP pour lui notifier la déconnexion du User en question.

Le serveur SIP consulte sa base de présence, qui voyait toujours le User comme connecté, et lui informe de la déconnexion de ce dernier en la demandant de mettre à jour les nouvelles informations. Les informations du User sont donc mis à jour, et il est donc absent de la base de présence. Si un message est envoyé en destination de ce User N, alors le serveur de détection de présence récupère le message, le retransmet au serveur SIP et celui-ci va tout simplement stocker le message dans la base de données MySQL en attendant la reconnexion du User N. La figure suivante montre que les deux user 1000 et 1001 sont bien connectés au serveur de SIP et un **Ping** leur est envoyé à chaque instant pour détecter leur présence.

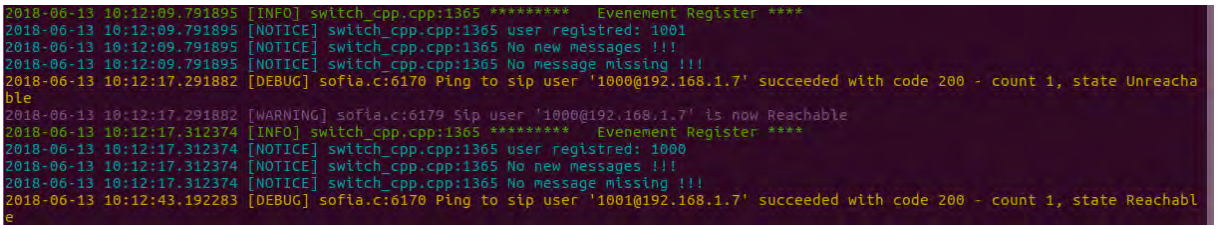

Figure 4.21: Connexion des user 1000 et 1001

Nous notons pour nos deux users les messages **succeeded** du **Ping** envoyé. Maintenant si le user 1000 désactive son WIFI tout simplement, notre **Ping** nous permet de savoir s'il est bien connecté ou pas. La figure suivante montre les messages de retour après le **Ping** envoyé au user 1000 qui désactive son WIFI.

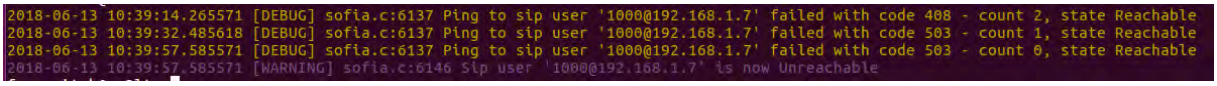

Figure 4.22: Message de retour après ping vers 1000

Le user 1000 est donc enlevé de la base de présence du serveur. Cela nous permet de savoir si un utilisateur est bien connecté ou pas afin de prendre des mesures par rapport aux messages qu'il reçoit. Si le user 1001 envoie un message au user 1000, le message est récupéré et stocké dans la base de données MySQL. La figure suivante montre le message envoyé au user 1000.

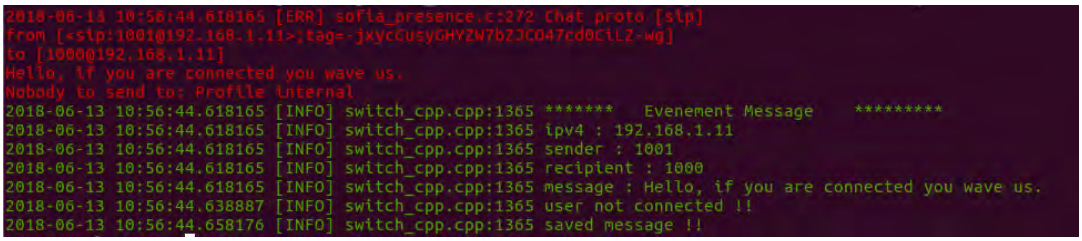

Figure 4.23: Message envoyé au user 1000 après notre modification.

Nous constatons que le message est bien sauvegardé dans la base de données pour attendre la bonne connexion du user 1000. La figure 4.24 suivante montre à la reconnexion du user 1000, le message qui était stocké lui est envoyé.

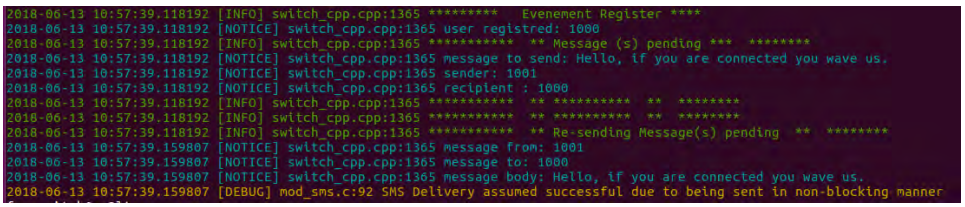

Figure 4.24: Message envoyé après reconnexion du user 1000

Un foi le user en question se reconnecte, il reçoit le(s) message(s) qui étaient stockés dans la base de données et qui avaient pour destinataire le concernant.

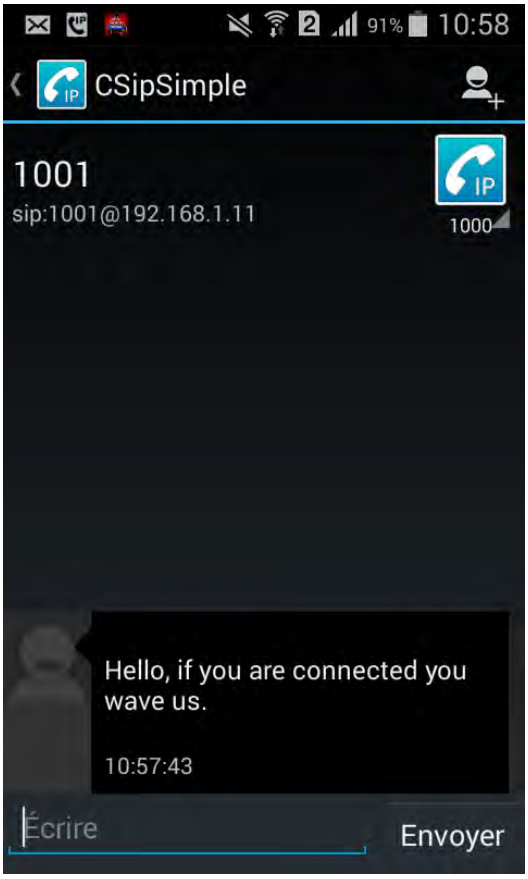

Figure 4.25: Message envoyé après reconnexion du user 1000

L'étude approfondi du protocole SIP concernant l'envoi et la réception des messages SIP a permis et constater des problèmes de réception du message envoyé au cas où le destinataire n'est pas connecté. Nous avons constaté que ce message est perdu et n'arrive jamais à destination dans ce cas. Cela a occasionné donc la solution que nous proposons dans cette proposition. Une solution que nous avons testée avec le serveur SIP freeswitch mais elle reste adapter à d'autres serveurs SIP comme astérisk. Aujourd'hui donc nous pouvons dire que cette solution va beaucoup contribuer à l'amélioration des conditions d'envoi et de réception des messages SIP et donc à une parfaite fiabilité des événements MESSAGES du SIP, ce qui est pour nous un avantage dans un contexte où nous tendons vers du tout IP avec l'arrivé de la quatrième génération (4G) et les perspectives d'une cinquième génération (5G).

4.4 CONCLUSION

Dans ce chapitre, nous avons mis l'accent sur les services de communication en temps-réel enrichis de données contextuelles recueillies dans les environnements des utilisateurs.

Ensuite nous avons proposé une nouvelle architecture pour le e-santé et son amélioration avec une couche de sécurité prenant en compte la confidentialité, l'intégrité, l'authenticité et un contrôle d'accès granulaire. Nous avons montré la faisabilité de notre approche en mettant en œuvre une preuve de concept. Ensuite nous avons proposé une solution pour améliorer la base de présence des plateformes utilisant le protocole SIP.

En outre, cette architecture apparaît comme un paradigme utilisable dans de nombreux cas d'utilisation dans les villes intelligentes, où les communications multimédias en temps réel doivent être couplées à des flux de données associés à des dispositifs IoT / WoT et IMS. Dans le chapitre suivant, nous allons proposer une solution permettant d'accéder à plusieurs Middlewares IoT intégrant du WebRTC à l'aide de SDN.

5 Accès à plusieurs Middlewares IoT intégrant du WebRTC à l'aide de SDN

5.1 Introduction

Dans la contribution précédente, les différents défis pour l'utilisation d'un middleware IoT à un seul espace intelligent ont été discutés. Ainsi, la nouvelle vision de la ville intelligente vise à fournir différents services à leurs citoyens afin d'améliorer leur qualité de vie. Cela implique systématiquement la présence de plusieurs espaces intelligents dans toute la ville et dans différents réseaux. Dans ce cas, le principal défi consiste donc à gérer tous ces espaces intelligents de manière efficace.

Dans ce chapitre nous décrirons d'abord une solution de middleware IoT pour le e-formation et e-Agriculture intégrant WebRTC et WoT. Ensuite, une architecture fonctionnelle qui intègre SDN dans une plate-forme IMS-IoT pour l'automatisation de la couche contrôle (plate-forme IMS-IoT) et de la couche transport est proposée. Enfin, cette architecture est ensuite illustrée dans les domaines de la santé en ligne, du e-learning et du e-agriculture.

5.2 Proposition d'une solution de middleware IoT pour le e-learning et e-agriculture

Pour rendre beaucoup plus efficace la formation dans le domaine des STEM en général et la télémédecine en particulier, nous avons proposé dans cette section une plateforme KMS-IoT permettant aux étudiants en télémédecine de collaborer avec leurs enseignants en vue de diagnostiquer des patients à distance à l'aide des objets connectés.

La plateforme KMS-IoT proposée peut être adaptée et aussi utilisée comme moyen de collaboration à distance entre apprenants et enseignants dans le secteur de l'agriculture de précision notamment en ce qui concerne la prise en compte des diagnostiques des maladies liées aux plantes cultivées ou la gestion de lutte contre les ravageurs des plantes cultivées.

Une vue simplifiée de l'architecture de notre plate-forme est illustrée à la Figure 5.1. À l'aide d'un navigateur compatible WebRTC, un utilisateur peut, d'une part, communiquer en temps réel avec un homologue distant et, d'autre part, accéder à son espace intelligent (e-learning ou e-agriculture). Conformément à l'approche WoT, l'application cliente doit prendre en charge à la fois la communication et l'accès aux objets intelligents. Chaque objet intelligent est censé

être identifié à l'aide d'un URI unique et offre une API Web permettant d'accéder à ses ressources. Les objets intelligents sont situés dans le réseau privé de chaque utilisateur (université, bureau, etc.). Ces objets ont leur propre organisation, que nous avons définit comme un espace intelligent.

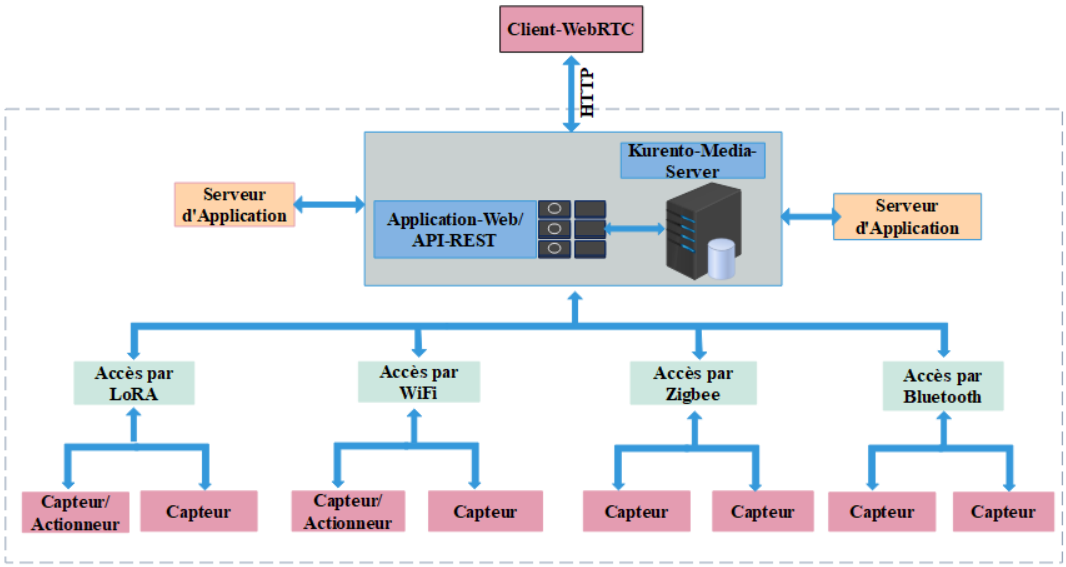

Figure 5.1: Architecture générale du middleware WebRTC-WoT

Pour la suite, nous présentons une description de notre architecture, dans laquelle sont définis les différents éléments impliqués ainsi que la manière dont ils communiquent. Les interactions principales et composants principaux de l'architecture :

1) Le « client WebRTC » situé dans le navigateur ;
2) La "passerelle" ;
3) Et enfin les objets intelligents, qui peuvent être des périphériques actifs ou passifs.

Le composant principal de l'architecture générale du middleware WebRTC-WoT est l'application web, l'API REST et le serveur multimédia Kurento. Le plus souvent les applications WebRTC traditionnelles sont normalisées, de sorte que les navigateurs puissent communiquer directement sans la médiation d'infrastructures tiers. Cela suffit pour offrir des services multimédias de base, mais des fonctionnalités telles que les communications de groupe, l'enregistrement de flux, la diffusion ou le transcodage sont difficiles à mettre en œuvre. Pour cette raison, les applications les plus intéressantes nécessitent l'utilisation d'un serveur multimédia.

Ainsi, la plupart des technologies de communication multimédia repose sur deux couches permettant d'abréger les fonctions clés dans tous les systèmes de communication interactive. Les deux couches sont le plan de signalisation et le plan média.

Le Plan de Signalisation est constitué par les parties du système chargées de la gestion des communications, c'est-à-dire les modules fournissant des fonctions de négociation de média, de paramétrage de la qualité de service, d'établissement d'appel, d'enregistrement d'utilisateur, de présence d'utilisateur, etc.

Plan média est formé par des fonctionnalités qui prennent en charge le traitement de média telles que le transport de média, le codage / décodage de média et le traitement de média.

La figure suivante montre une représentation conceptuelle de l'architecture de haut niveau de Kurento:

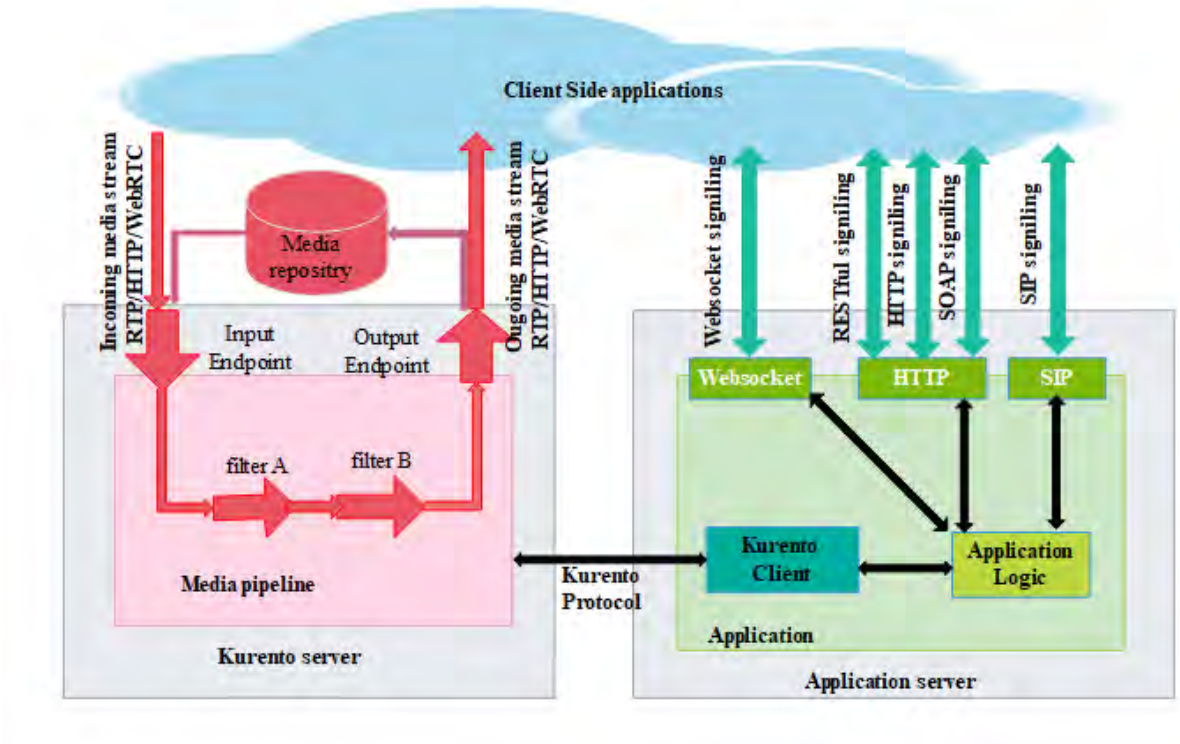

Figure 5.2: Architecture de signalisation et média de Kurento

Pour la suite nous allons discuter sur une proposition qui nous permet de contribuer à l'amélioration de l'enseignement à distance de la médecine et du e-agriculture en proposant une plateforme de travaux pratiques

5.2.1 Scénario pour e-learning et e-agriculture

Plusieurs approches ont été proposées pour rendre disponibles les travaux pratiques dans les formations e-learning. Mais, il existe des disciplines en STEM telles que la médecine et la biologie où il est difficile de faire du e-learning en raison du manque de plateformes et d'équipements de laboratoire à distance. L'enseignement à distance de la médecine est confronté à de multiples difficultés techniques et logistiques. En effet, les étudiants sont géographiquement dispersés et ne disposent pas de laboratoires virtuels pour les Travaux Pratiques.

5.2.1.1 Scénario E-learning

La solution proposée permet au professeur d'initier des Travaux Pratiques (TP) de consultation médicale (par exemple) en direct via la plateforme Kurento Media Server Internet des objets (KMS-IoT). L'objectif du TP est d'apprendre aux étudiants comment se déroule un examen clinique par exemple avant d'aller en stage pratique. Pour ce faire, le professeur utilise les appareils médicaux connectés (stéthoscope, thermomètre, électrocardiogramme, tensiomètre, etc.) pour prendre les constantes d'un étudiant pris comme patient. Ces données sont directement transmises à l'application puis partagées en temps réel avec les autres étudiants via la plateforme. A l'aide d'un ordinateur, d'une tablette ou d'un smartphone connecté, n'importe quel étudiant peut visualiser les constantes et suivre en temps réel les commentaires/explications de l'enseignant. Chaque étudiant peut également interagir en posant des questions. La figure 5.3 décrit l'architecture du système proposé.

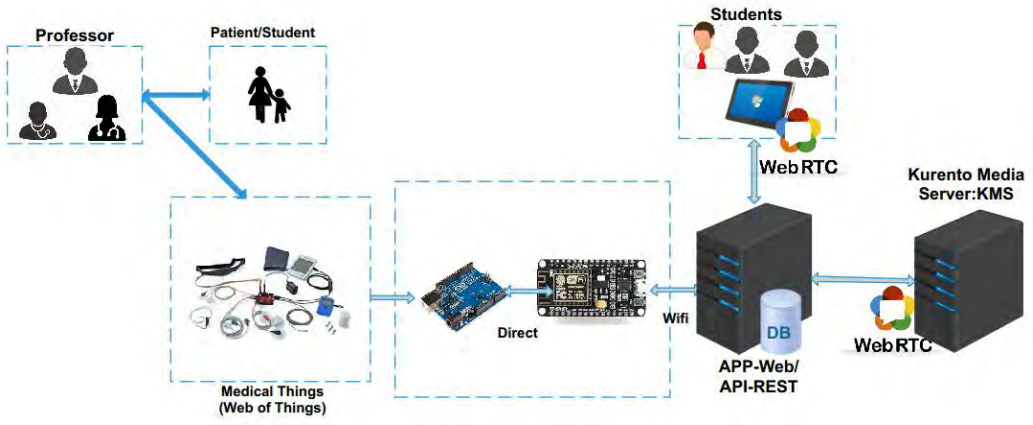

Figure 5.3: Architecture du système proposé

5.2.1.2 Scénario E-agriculture

L'objectif de cette section est de décrire la contribution qui vise à améliorer l'enseignement à distance dans les secteurs agricoles en proposant une plateforme collaborative permettant de

faire des visites de terrain virtuelles et même de partager de ressources. Pour ce faire, nous couplons l'intelligence du Web of Things (WoT) à la puissance du serveur multimédia WebRTC Kurento.

L'enseignement à distance dans le domaine de l'agriculture, la biologie, etc. est confronté à de multiples difficultés techniques et logistiques. En effet, les visites de terrains représentent un complément indispensable du cours théorique dispensé aux étudiants. Il arrive que les zones abritant la biodiversité recherchée soient instables et potentiellement dangereuses pour les non-résidents. La solution proposée permet aux étudiants des instituts agricoles supérieurs de réaliser une excursion pédagogique visant à étudier la biodiversité d'une localité comme par exemple le sud du Sénégal. Pour ce faire, un groupe résident dans la région est sélectionné pour la prise des mesures (humidité du sol, composition chimique, température ambiante, etc.) sous la supervision d'un tuteur. Les données collectées sont directement transmises à l'application puis partagées en temps réel avec les autres étudiants via la plateforme. À l'aide d'un ordinateur, d'une tablette ou d'un smartphone connecté, n'importe quel étudiant peut visualiser les mesures et suivre en temps réel les commentaires/explications du tuteur. Chaque étudiant peut également interagir en posant des questions.

Dans la littérature, des travaux ont montré qu'on peut utiliser l'agriculture électronique de précision (e-precision agriculture) pour améliorer l'arrosage, la fertilisation des sols et le rendement des récoltes. Les auteurs de [198] [199] [200] proposent des solutions basées sur arduino et les capteurs (pH sensor, water level sensor, servo…) pour économiser l'utilisation de l'eau et des engrais sur de petites surfaces cultivables. Ceux de [201] ont mis en place un système de transmission basé sur LoRa et bluetooth pour récupérer les informations des capteurs (température, d'humidité, de CO_2, luminosité) dans un champ. Ces informations sont stockées dans une base de données et peuvent être consultées à l'aide d'un smartphone. Dans les articles [202] [203], les auteurs proposent un site web permettant de réguler la vente des produits agricoles. Les auteurs de l'article [204] proposent un module d'apprentissage en ligne pour former les agriculteurs sur le bon choix des semences, la lutte antiparasitaire, les effets secondaires de l'utilisation des pesticides, etc. Malgré la pertinence de ces travaux, peu de solutions sont axées sur la formation à distance des étudiants des filières agricoles.

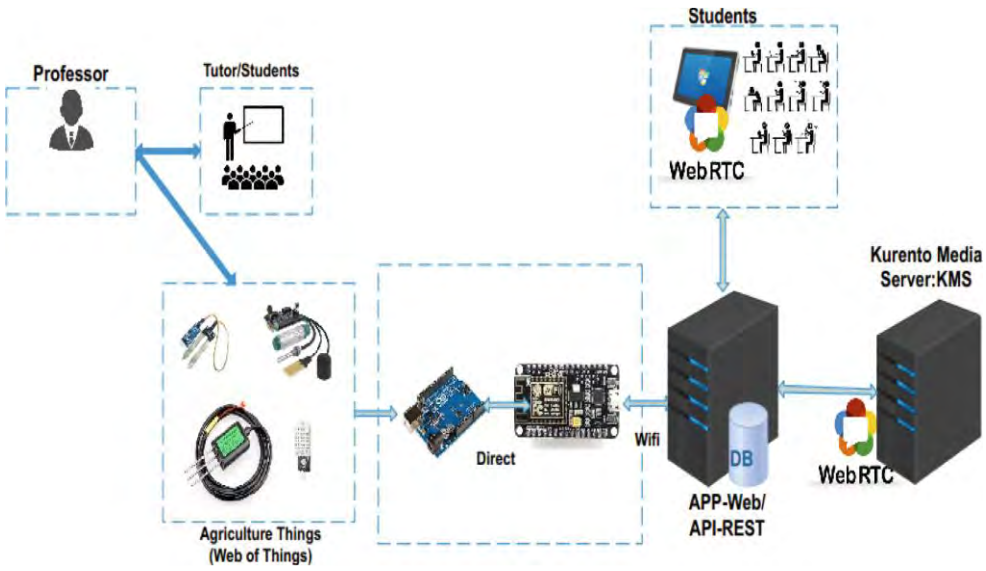

Figure 5.4: Architecture du système proposé

5.2.2 Implémentation

Pour implémenter une telle architecture, nous avons créé un Media Pipeline composé de 1+N WebRtcEndpoints. Le flux du « Presenter » est diffusé au reste des « Viewers ». Les étudiants accèdent au flux avec l'id viewer. Le client et le serveur d'application communiquent en utilisant un protocole de signalisation basé sur des messages JSON.

Une plateforme utilisant Node.js et Kurento Media Server (KMS) est implémentée dans cette proposition. D'une part, elle permet d'établir une communication multimédia entre deux utilisateurs en utilisant simplement leur navigateur. D'autre part, elle permet aux utilisateurs d'accéder aux données des objets connectés prédéfinis. Une description des entités de l'implémentation est faite dans le chapitre 4. L'architecture proposée est composée de trois entités distinctes :

Internet des objets : Chaque point de terminaison est considéré comme une passerelle vers son ensemble d'objets intelligents. En outre, chaque utilisateur a un contrôle sur ces objets. Le nœud d'agrégation (Raspberry Pi, NodeMCU ESP8266 etc.) n'est pas responsable de la lecture des capteurs. Il fournit simplement une passerelle entre l'utilisateur et le réseau de capteurs, puis effectue des analyses de données. Le nœud de capteurs est le niveau le plus bas d'un réseau de capteurs. Il est chargé de recueillir des informations auprès des capteurs, d'effectuer des actions de l'utilisateur et d'utiliser des mécanismes de communication pour envoyer des données au nœud d'agrégation.

En utilisant l'architecture actuelle, la mise en œuvre d'une sortie pédagogique à distance est possible, où un tuteur communique avec les étudiants en utilisant Kurento Media Server.

Dans le cas des scénarii d'e-santé, tout ce dont nous avons besoin que des capteurs médicaux portables. Ils peuvent communiquer via n'importe quel protocole, puisque le WoT résume toute la complexité de la connectivité des objets.

API : Nous avons développé une API REST capable de récupérer les informations collectées par un appareil médical connecté et de les stocker dans une base de données MongoDB. MongoDB appartient à la famille NoSQL Document-store, développée en C++. Il est basé sur le concept de pair clé-valeur. Le document est lu ou écrit en utilisant la clé. MongoDB prend en charge les requêtes dynamiques sur les documents. Comme il s'agit d'une base de données orientée document, les données sont stockées sous la forme de JSON, style BSON [205] ;

Application web : Pour mettre en place l'application web, nous utilisons les technologies NodeJs et Kurento Media Server. Cette plateforme permet aux enseignants et aux étudiants de s'enregistrer puis de s'authentifier afin d'accéder aux fonctionnalités de Kurento Media Server. Une fois connectés, les étudiants peuvent visualiser les données des capteurs et les flux médias du tuteur en charge de la sortie pédagogique. La figures 5.5 montre le principe d'authentification sur la plateforme KMS-IoT-E-Agriculture.

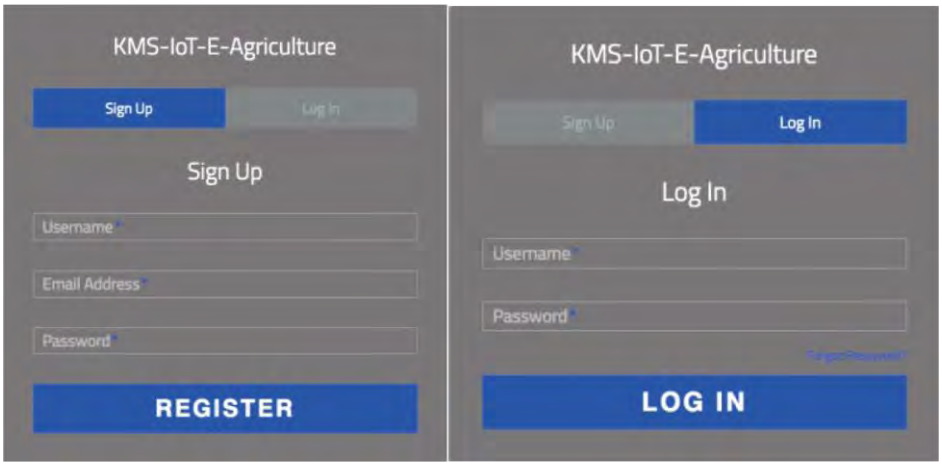

Figure 5.5: Création et connexion des utilisateurs sur KMS-IoT-E-agriculture

L'application web permet également de recueillir des informations de la base de données et de les afficher. Les utilisateurs connectés peuvent alors visualiser les données des capteurs. La figure 5.6 montre que les acteurs peuvent accéder aux informations du capteur de température et d'humidité. Le même mécanisme est applicable à tout autre capteur.

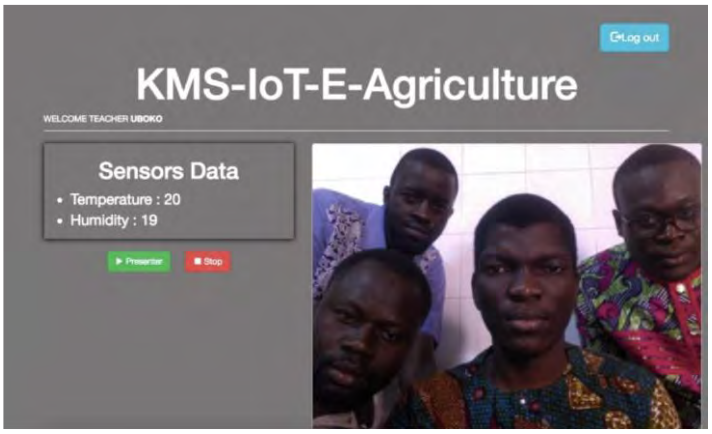

Figure 5.6: Communication entre professeur et étudiant coté professeur

Dans ce cas le professeur à le rôle de présentateur, c'est-à-dire il est le seul à avoir la possibilité de publier les données du capteur sur la plateforme. Ainsi un étudiant qui s'est bien enregistré aura la possibilité de visionner les données publiées par l'enseignant en charge de l'excursion ou de l'examen clinique dans le cas de la télémédecine. La figure 5.7 nous montre l'interface de connexion d'un étudiant pour visualiser les données des capteurs de température et d'humidité.

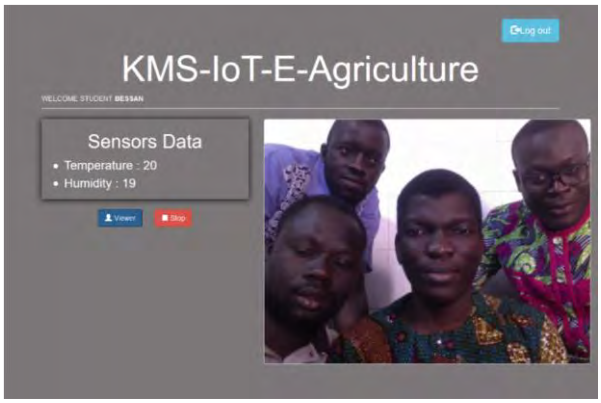

Figure 5.7: Communication entre enseignant et étudiants : du côté des étudiants

La plateforme KMS-IoT-Agriculture permet de faire des TP dans l'enseignement à distance de l'agriculture. Cette plateforme est bâtie autour d'un serveur multimédia WebRTC Kurento, d'une API, d'une application Web et d'objets connectés. Elle permet à des étudiants en formation à distance de réaliser une excursion virtuelle visant à étudier la biodiversité d'une localité. L'adoption de l'approche proposée pourrait contribuer à l'amélioration de l'enseignement à distance dans le domaine agricole.

5.3 Automatisation des entités d'une ou de plusieurs middleware(s) IoT basée sur SDN

Cette section propose la dernière contribution et est structuré comme suit. Il commence par une présentation globale et générale des principaux défis abordés dans cette partie. Ensuite, fournit les travaux connexes des architectures qui proposent un couplage entre le WoT / IoT et le réseau défini par logiciel (SDN). La solution implémentée est décrite avec une explication des différents composants et des différentes interactions, suivie des détails techniques relatifs à la mise en œuvre. Cette architecture est ensuite illustrée dans les domaines de e-santé, de e-formation et de e-agriculture. Enfin, une architecture fonctionnelle qui intègre SDN dans une plate-forme IMS-IoT pour l'automatisation de la couche contrôle (plate-forme IMS-IoT) et de la couche transport est proposée. Cette contribution permet de bénéficier de toute la simplicité et l'efficacité fournies par un réseau SDN mais également des services offerts par la plate-forme IMS-IoT-KMS.

5.3.1 Défis et enjeux

L'IoT est une chance pour les données réelles, d'où l'importance de ne pas ignorer la place du réseau informatique et de télécommunication pour son déploiement. Ce qui constitue une chance pour se rattraper dans les domaines e-santé, e-learning et e-agriculture pour les pays en voie de développement. Suivant les données massives reçues des capteurs l'administrateur réseau peut être confronté à un problème, donc la tendance est de faire appel au réseau programmable (SDN) pour automatiser certaines taches devenues difficiles à administrer afin de faciliter l'accès à plusieurs espaces intelligents.

Commençons d'abord par définir le problème principal à résoudre. L'objectif principal est de fédérer l'accès à tous les Smart Objets appartenant à une entité donnée (qui peut être un utilisateur, un fournisseur de services, une infrastructure, etc.) mais aussi d'automatiser les couches contrôle et transport. Par exemple, un utilisateur peut avoir un Smart Space dans le E-santé, un deuxième Smart Space dans le E-Agriculture et un autre dans le E-learning, comme illustré à la Figure 5.8.

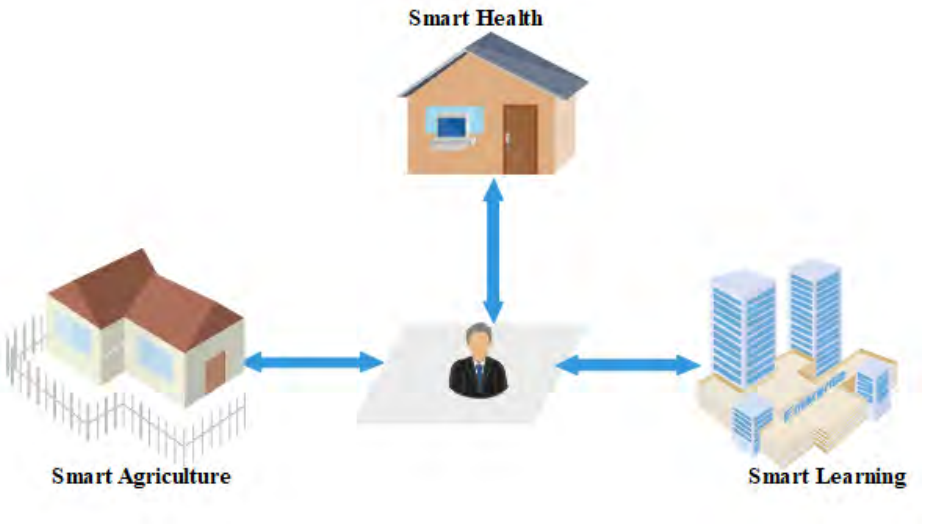

Figure 5.8: Architecture de plusieurs espaces intelligents appartenant à une même entité

Sachant que tous ces espaces intelligents appartiennent à la même entité et que leur nombre peut varier d'un cas d'utilisation à un autre. La gestion de chaque Smart Space séparément est une tâche ardue pour un administrateur. De plus, l'administrateur doit faire preuve de prudence lors de la mise à jour des stratégies d'accès dans chaque Smart Space afin de garantir leur cohérence, pour éviter les conflits, etc.

La qualité de service d'applications IoT dépend non seulement du réseau Internet et de l'infrastructure de communication, mais aussi du fonctionnement et des performances des appareils IoT. Par conséquent, les nouveaux paramètres de QoS tels que la précision des données et la disponibilité des appareils deviennent importants pour les applications IoT par rapport aux applications Internet.

Une solution consiste à une automatisation de la couche contrôle et la couche transport afin de gérer la QoS de tous ces espaces intelligents à l'aide d'une entité centralisée. Cette entité doit pouvoir garantir toutes les exigences de qualité de services sur l'accès à l'un des espaces intelligents, tout en permettant l'établissement des routes sécurisées entre l'utilisateur et l'espace intelligent demandé.

Une solution triviale peut consister à utiliser un serveur centralisé, qui contrôle tous les espaces intelligents. Ce serveur devra gérer l'authentification de l'utilisateur demandeur, la gestion des stratégies de contrôle d'accès de tous les espaces intelligents et la création de canaux sécurisés d'abord entre le serveur et les espaces intelligents et entre l'utilisateur et l'espace intelligent.

L’inconvénient majeur de cette solution est que, dans cette vision, tous les espaces intelligents doivent être exposés à Internet à l’aide d’adresses IP publiques, ce qui ajoute une nouvelle surface d’attaque pour les pairs malveillants. Aussi, deux canaux sécurisés doivent être établis : Un entre le serveur et l'espace intelligent et un autre entre l'espace intelligent et l'utilisateur. Enfin, le serveur ne peut pas gérer le routage et configurer les équipements réseau, tels que les pare-feu, au niveau de la couche réseau entre l'utilisateur et l'espace intelligent.

Par conséquent, pour conserver l’avantage d’avoir un point de décision centralisé ainsi qu’une solution pour gérer les problèmes liés au réseau, tels que le routage, et éviter les problèmes mentionnés précédemment, nous proposons l’utilisation d’un contrôleur SDN (Software Defined Networking).

SDN a suscité un vif intérêt de la part des milieux universitaires et industriels. Cela change la manière traditionnelle de gérer le réseau, où les décisions de routage sont prises par le équipements de réseau eux-mêmes (les routeurs). Il est un nouveau paradigme pour les réseaux informatiques apparu récemment pour cacher la complexité de l’architecture de réseau traditionnelle (par exemple de l’Internet) et briser la fermeture des systèmes de réseau dans les fonctions de contrôle et de données. Il permet aux propriétaires et aux administrateurs de réseau de contrôler et de gérer le comportement du réseau par programme, en découplant le plan de contrôle du plan de données. SDN a le potentiel de révolutionner les réseaux informatiques classiques existants, en offrant plusieurs avantages tels que la gestion centralisée, la programmabilité du réseau, l’efficacité des coûts d’exploitation, et les innovations [206].

Ainsi, en disposant d'un point de décision centralisé pour le contrôle d'accès directement lié au contrôleur, ainsi que du mécanisme d'authentification, il est possible de gérer de manière dynamique et efficace la QoS des différents Smart Spaces. De plus, puisque le contrôleur SDN sait comment router les différents paquets vers l'espace intelligent correspondant, il n'est pas nécessaire que l'espace intelligent soit directement exposé à Internet.

Pour avoir une meilleure qualité de service : Pouvoir gérer efficacement le trafic de données pour faciliter la mise en œuvre de la qualité de service (QoS). Par exemple, dans le cas des transmissions voix sur IP et multimédia, une plus grande largeur de bande peut être allouée pour ces services particuliers à l'aide du contrôleur. Par conséquent, assurer une meilleure qualité de service. Un exemple pratique peut être lié à WebRTC, où, par exemple,

un client doit utiliser une caméra située dans l'un des Smart Spaces appartenant à son domaine afin de communiquer en audio / vidéo avec un autre utilisateur. Le SDN facilitera l'accès à ce Smart Space et à cette caméra et allouera plus de bande passante au flux multimédia afin d'obtenir une meilleure qualité de service.

Pour faciliter la connectivité des utilisateurs, en créant des itinéraires directs et sécurisés vers la ressource demandée. La sécurité aux couches de bas niveau concerne principalement les règles de pare-feu relatives au contrôle du trafic, tandis que la sécurité aux couches de haut niveau consiste en l'authentification, l'autorisation, le contrôle d'accès, la confidentialité et l'intégrité.

Tableau 5.1: Comparaison entre une architecture n'utilisant pas le SDN et une autre utilisant SDN

	Sans SDN	Avec SDN
Confidentialité	OUI	OUI
Intégrité	OUI	OUI
Authentification	OUI	OUI
Contrôle D'accès	OUI	OUI
Contrôle trafic(3)	NON	OUI
Management DoS(2)	NON	OUI
Contrôle d'application(1) (4)	NON	OUI
Qualité de services	NON	OUI
Ménagement dynamique des équipements Réseaux	NON	OUI
Connecter le client à la passerelle	difficile	facile
évolutivité	NON	OUI

(1) Gestion du trafic inconnu avec des règles strictes (les bloquant éventuellement par défaut).

(2) Identifier les attaques DoS et les prévenir en contrôlant les flux de données.

(3) Identifier et contrôler les applications qui écoutent sur n'importe quel port, pas

seulement les standards.

(4) La possibilité de contrôler les flux et la sécurité des différentes applications interagissant avec le système de manière simple.

5.3.2 Architecture globale

En règle générale, l'utilisation d'un contrôleur SDN présente plusieurs avantages, notamment :

- **L'Approvisionnement centralisé du réseau :** afin d'avoir une vue centralisée de l'ensemble du réseau, ce qui facilite la centralisation de la gestion et de l'approvisionnement des équipements du réseau ;
- **L'Interaction avec la couche d'application :** permet aux applications d'émettre des commandes et des demandes au contrôleur et de fournir des fonctionnalités supplémentaires qui ne peuvent pas être effectuées par le contrôleur ;
- **Une Sécurité plus fine :** un point de contrôle central sur la topologie globale, qui peut souvent changer, peut faciliter la gestion des stratégies de sécurité. En plus de pouvoir disposer de politiques de sécurité cohérentes sur toute la topologie. En effet, la centralisation du contrôle de la sécurité dans un seul contrôleur SDN présente le désavantage de créer un point de défaillance unique. Cependant, ces problèmes peuvent être résolus en ayant une implémentation sécurisée et appropriée du SDN et en ayant plusieurs instances du contrôleur.

Ainsi, afin de résoudre les problèmes mentionnés précédemment, en particulier ceux de QoS, une architecture WoT-SDN est proposée. En outre, le recours à l'API du contrôleur SDN en direction du nord devrait influer sur le déploiement à grande échelle du SDN, où plusieurs applications « tierces » peuvent s'interfacer avec l'interface en direction du nord afin de fournir davantage de services, notamment en ce qui concerne la sécurité et la gestion de la vie privée.

L'architecture peut être divisée en deux parties principales : Les entités qui interagissent avec l'interface nord du contrôleur. Et les entités qui interagissent avec l'interface sud du contrôleur.

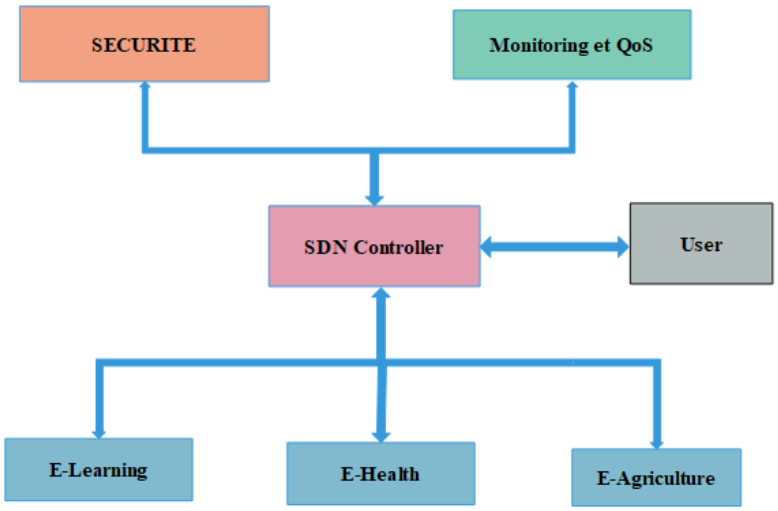

Figure 5.9: L'architecture globale avec le contrôleur SDN

La première catégorie représente les différents mécanismes complémentaires aux composants de base du SDN. Ils ajoutent principalement des mécanismes de sécurité, de surveillance et de qualité de service au système :

- **Les mécanismes de sécurité** sont composés d'un tiers authentifiant, d'un contrôle d'accès et d'une PKI pour gérer les clés de chiffrement. Les mécanismes de surveillance pour contrôler le trafic et déployer éventuellement des mesures de détection supplémentaires, tels qu'un IDS, un IPS, SEM, etc. ;
- **Les mécanismes de surveillance :** afin de surveiller le trafic et éventuellement de déployer des mécanismes de détection de sécurité supplémentaires, tels que IDS ou IPS, etc. ;
- **Les mécanismes de QoS** permettent de gérer la QoS des différentes Smarts Spaces. Par exemple, pour allouer plus de bande passante à certains Smart Objets pour un cas particulier.

Et la deuxième catégorie représente les clients et les différents Smart Spaces. Le point d'entrée de chaque Smart Space est une passerelle, qui expose une API RESTful afin d'interagir avec les différents Smart Objets.

5.3.3 Architectures SDN-IoT / WoT

Cette section présente d'abord les travaux des architectures associant le SDN au WoT / IoT en général, à la santé en ligne, au e-learning et e-agriculture en particulier, afin de gérer

plusieurs Smart Spaces, ainsi qu'une comparaison avec notre proposition. Ensuite, fournit un aperçu rapide lié à l'utilisation du SDN pour gérer l'automatisation de la couche d'accès, du transport et la QoS du réseau et des infrastructures ou middleware IoT.

Plusieurs travaux ont été proposés dans la littérature pour les architectures combinant les capacités de contrôle SDN et les avantages apportés par les paradigmes WoT / IoT. Les auteurs dans [207] proposent d'examiner les dispositifs terminaux sans capacités de routage ni de commutation, mais uniquement la transmission et la réception de données, en plus des éléments de réseau traditionnels tels que les routeurs et les commutateurs. Ils proposent de déléguer le contrôle et la gestion du réseau en connectant de nombreux périphériques d'un environnement intelligent à une entité logicielle, tout en laissant le contrôle des périphériques du réseau à l'utilisateur final. Ils introduisent la notion de SD-LAN (réseau défini par logiciel) afin de créer un environnement virtuel pouvant regrouper plusieurs objets provenant de réseaux différents comme s'ils se trouvaient sur le même réseau local. Les auteurs dans [208] proposent de construire l'architecture de WoT orientée ressources basée sur SDN. Selon eux, l'utilisation du SDN offre une nouvelle possibilité d'améliorer les performances, de simplifier la gestion et d'améliorer la QoS et la sécurité de WoT. Ils proposent de résoudre les problèmes liés à la faiblesse des fonctionnalités de gestion, au manque de mécanismes d'accès efficaces et aux conflits entre l'ouverture requise par le WoT et la sécurité requise par les applications utilisant le SDN. Cependant, ils ne disposent pas de la partie contrôle d'accès. Dans leurs travaux de recherche O. Flauzac et al., présentent une architecture théorique combinant l'IoT et les capacités réseau du SDN afin de faire face aux défis de la sécurité du réseau. Ils introduisent la notion de domaine SDN, chaque domaine étant contrôlé par un seul contrôleur SDN [209]. Chaque domaine peut contenir des nœuds contraints ou non contraints avec suffisamment de ressources. Le périphérique / nœud contraint peut être associé à un autre voisin, qui dispose de suffisamment de ressources et de capacités SDN. Ils traitent également du cas de plusieurs domaines SDN et de leurs interactions. Pour cela, ils ont introduit la notion de « contrôleur de frontière » (BC) situé à l'extrémité de chaque domaine SDN. Ce BC sera alors responsable de l'interaction avec les autres BC des autres domaines. Cependant, la proposition n'est qu'une architecture théorique, ils manquent d'expérimentation et de résultats. De plus, ils ne traitent pas du contrôle d'accès aux ressources des périphériques. Les auteurs dans [210] proposent d'utiliser une passerelle SDN comme moyen distribué de surveiller le trafic en provenance / à destination des périphériques IoT. Cette passerelle est également capable de détecter des attaques et des comportements anormaux et éventuellement de les

contrer.

Les principaux défis de l'intégration du Software Defined Networking (SDN) dans une plateforme IMS-IoT et l'automatisation des couches contrôle et transport des données recueillies par les objets sont discuté par [211]. IP Multimedia Subsystem (IMS) est une technologie qui gère l'établissement, la gestion et la terminaison de session via le protocole SIP (Session Initiation Protocol) afin de permettre le déploiement de services sur tous les réseaux IP. IMS fournit une interface IP commune qui simplifie la signalisation et le développement d'applications et facilite le traitement de plusieurs sessions pouvant être utilisées par différentes applications. En outre, il permet le contrôle d'accès et la gestion de la facturation, ce qui permet une gestion cohérente des différentes applications sur le réseau [212] [213]. Les avantages de l'introduction de l'IMS dans le domaine IoT ont déjà été examinés par plusieurs équipes de recherche

5.3.4 Proposition d'une solution intégrant le SDN sur plusieurs Smarts Spaces

Dans cette section, nous proposons une nouvelle conception innovante d'un système de gestion de la qualité de service de l'architecture Web of Thing basée sur SDN. L'architecture proposée a pour objectif de traiter les problèmes et défis précédemment discutés dans la section 5.3.1. Par conséquent, pour gérer les différents aspects de la QoS telle que la vue d'ensemble de tous les nœuds du réseau permettant à l'administrateur de suivre plus facilement la fréquence à laquelle une seule connexion est utilisée. Il peut réagir en temps réel aux connaissances acquises et réguler le trafic de données en conséquence afin d'être en mesure de fournir à tout moment la bande passante promise à tous les participants.

5.3.4.1 Les outils et protocoles utilisés

Avant de commencer la partie technique, plusieurs hypothèses ont été considérées, à savoir :

- ✓ Les Smart Spaces sont déjà configurés, chaque Space intelligente ayant une passerelle avec une adresse IP privée fixe et connectée à l'ensemble des Smart Objects. Pour prouver notre concept, nous avons mis en œuvre trois Smart Spaces. Cependant, l'hypothèse concerne principalement le déploiement à grande échelle de l'architecture avec plus de Smart Spaces;
- ✓ Les utilisateurs et leurs Smarts Spaces correspondants sont également configurés à

l'avance. Un administrateur peut configurer ses profils et ses Smarts Spaces à l'aide de l'interface utilisateur de l'application ;

- ✓ Les clés symétriques sont pré-partagées entre le serveur et les passerelles ;
- ✓ Les principaux composants sont situés dans un réseau sécurisé tel que le fournisseur de services Internet, qui peut être celui qui fournit la proposition à ses clients (utilisateurs privés, entreprises privées, infrastructures intelligentes, etc.).

❖ **Réseau Mininet**

La figure suivante représente la topologie Mininet simulant les différentes parties réseau de notre architecture. Il contient principalement cinq OpenFlow Switch (OFS). Trois d'entre eux sont directement reliés à une passerelle physique (une raspberry pi), via l'un des ports OFS, représentant ainsi les trois Smarts Spaces (e-santé, e-formation et e-agriculture). L'un est directement lié aux utilisateurs. Et on crée une séparation pour le réseau, qui peut être vu comme un routeur. Tous les OFS sont liés au contrôleur OpenSDNCore et communiquent à l'aide du protocole OpenFlow. L'OFS peut également être configuré afin de créer des VLAN correspondant à chaque Smart Space (e-health ou e-learning ou e-agriculture par exemple).

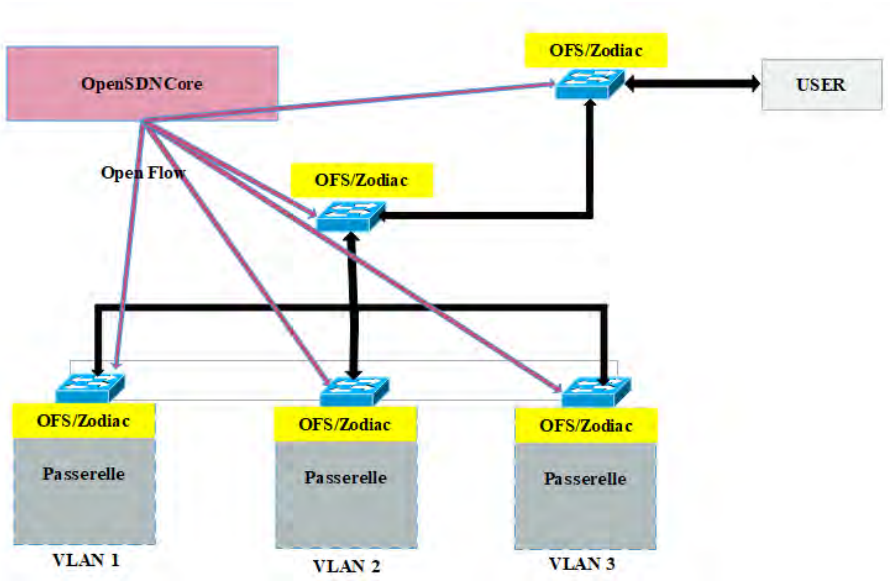

Figure 5.10: Construction de la topologie du réseau avec Mininet

❖ **OpenSDNCore**

OpenSDNCore est un contrôleur basé sur OpenFlow qui crée un intermédiaire entre les équipements réseau et la couche application et gère l'ensemble du réseau. Le contrôleur est basé sur une architecture à trois niveaux illustrée à la Fig. 5.11. Les trois composants clés du contrôleur sont les suivants : (1) la couche de fonction principale, (2) l'interface en direction

du nord et (3) l'interface en direction du sud.

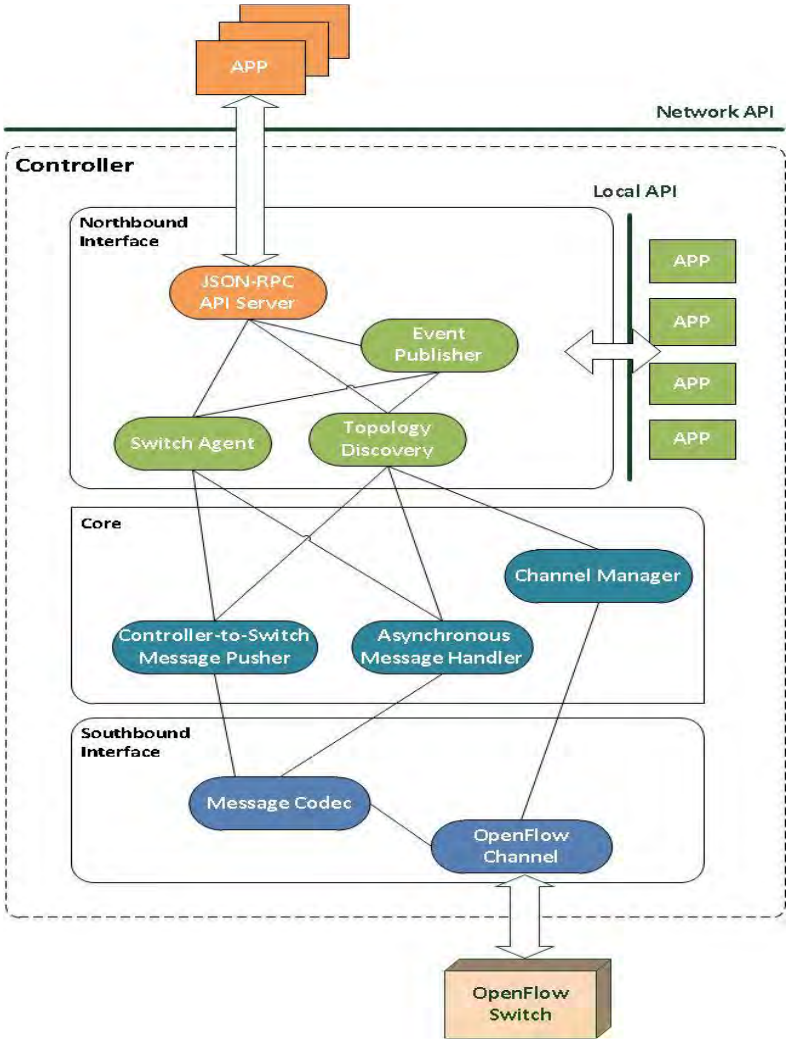

Figure 5.11: Composants clés d'OpenSDNCore [214]

L'interface en direction du sud gère la communication avec les systèmes OFS à l'aide du protocole OpenFlow. Il se compose de deux composants : le canal OpenFlow et le codec de message OpenFlow. Le canal OpenFlow connecte chaque commutateur au contrôleur. Le codec de message OpenFlow est utilisé pour coder / décoder chaque fois qu'un message est envoyé ou reçu.

La couche de fonction principale contient un ensemble réduit de fonctions requises pour le déploiement du contrôleur. Le contrôleur pour basculer le message construit les nouveaux messages et les envoie au canal OpenFlow. Pour les messages asynchrones et symétriques provenant du commutateur, le gestionnaire de messages correspondant sera appelé, ce qui traitera le message et le distribuera aux fonctions de l'interface Northbound ou

d'autres applications définies par l'utilisateur en dehors de l'espace principal.

L'interface Northbound est le facteur clé de la programmabilité du réseau. La conception ouverte présentée permet aux développeurs de réseaux de développer des applications en utilisant deux types d'API différents exposés par le contrôleur. Le composant clé est le serveur JSONRPC, qui est un protocole d'appel de procédure distante et léger sans état. Il intercepte les requêtes provenant des différentes applications et exécute les commandes correspondantes. Il utilise une version étendue de JSON comme format de données.

Avec l'API locale, les applications sont hébergées sur le même ordinateur que le contrôleur et utilisent le même langage de programmation. Cependant, il n'est pas toujours le cas de conserver l'application et le contrôleur dans le même environnement. Ainsi, l'API réseau offre la possibilité de communiquer à distance. Le contrôleur exécute un serveur API pour gérer les demandes entrantes provenant d'autres applications réseau.

Afin de mettre en œuvre notre approche, trois principaux composants supplémentaires ont été développés et ajoutés à l'API JSON RPC : (1) « Découverte ARP », qui permet au contrôleur de localiser l'emplacement des passerelles uniquement en fonction de son adresse IP, (2) « Créer Route "qui permet à l'utilisateur de communiquer avec la passerelle sélectionnée et enfin (3) " Envoi de paquet à passerelle "qui permet au serveur d'envoyer tout type de données à la passerelle sélectionnée. Dans les sections suivantes, nous examinerons chacune d'elles en détail.

- **Arp discovery**

Le contrôleur doit avoir une vue globale de la topologie globale. Lorsque le contrôleur est lancé pour la première fois, il ne connaît pas la topologie de réseau des réseaux locaux impliquant les différentes passerelles représentant les différents Smarts Spaces. Le moyen le plus simple était de les envoyer manuellement. Cependant, pour automatiser la procédure, une méthode d'envoi de requêtes ARP aux différentes passerelles a été implémentée. Ainsi, pour chaque passerelle, le serveur a envoyé la commande correspondante et en tant que paramètres, l'adresse « IP source », qui est une adresse IP aléatoire du même réseau que la passerelle, et l'adresse « IP de destination », qui est l'IP de la passerelle. Le contrôleur forge alors un paquet ARP en fonction des informations fournies. Ce paquet sera encapsulé dans un paquet IP, puis à nouveau dans un paquet Ethernet. Ce paquet sera ensuite transporté à l'aide d'un message OpenFlow appelé «Packet Out» et sera envoyé à tous les commutateurs.

❖ **Module de routage de bout en bout**

Par mesure de sécurité et avant la création d'un canal sécurisé entre l'utilisateur et la passerelle, seuls les protocoles ICMP et ARP sont autorisés dans la topologie, afin d'empêcher les utilisateurs malveillants d'accéder aux passerelles et de réduire au minimum la surface d'attaque. Par conséquent, un module appelé « Routage de bout en bout » est mis au point afin de permettre à l'utilisateur légitime d'accéder aux passerelles mais aussi d'utiliser des protocoles supplémentaires tels que TCP et UDP. Cette méthode ajoute une route entre deux points d'extrémité à la table de routage des commutateurs d'apprentissage. L'avantage de ce module est que seules les adresses IP source et de destination sont nécessaires pour créer la route. La méthode fonctionne comme suit :

- À partir du serveur, une commande avec les deux adresses IP de l'utilisateur et la passerelle du Smart Space demandé pour créer un itinéraire est émise. Les deux adresses IP sont connues car l'adresse IP de la passerelle est statique (un DHCP peut être implémenté dans des travaux futurs), et l'adresse IP de l'utilisateur est connue du serveur, ce qui peut être extrait de la requête HTTPS ;
- Le serveur JSON RPC situé dans le contrôleur SDN et qui représente l'interface nord du contrôleur, reçoit la demande et vérifie le format de la demande. À partir des paramètres, il reçoit un objet JSON contenant les adresses source et de destination, qui peuvent être des adresses IPv4, IPv6 ;
- Le principe de fonctionnement de la fonction principale, « créer un itinéraire » :
- Prend en argument : l'adresse source, l'adresse de destination et les filtres s'ils existent ;
- Crée un pool de mémoire pour le module ;
- Appelle ensuite le "hosttracker", qui recherche l'hôte, avec l'adresse IP passée dans l'argument, dans toute la topologie. Hosttracker renvoie toutes les informations relatives à l'hôte, telles que l'adresse IP, l'adresse MAC, le port sur lequel l'hôte est connecté et l'ID OpenFlow Datapath (DPID) du commutateur directement lié ;
- Hosttracker est appelé deux fois dans ce cas, pour l'hôte source et pour l'hôte de destination ;
- Avec ces informations, la topologie peut être analysée afin de rechercher le chemin le plus court entre le DPID du commutateur directement lié aux hôtes (c'est-à-dire la passerelle et l'utilisateur). L'algorithme utilisé ici pour rechercher le plus court chemin est Dijkstra [215] ;
- De nouveau, la recherche du chemin le plus court est appelée deux fois, d'abord pour

rechercher le chemin le plus court de la source à la destination. Et pour rechercher le chemin le plus court de la destination à la source, car il peut changer ;

- Les deux routes avec les adresses IP correspondantes sont ensuite poussées du contrôleur vers les commutateurs en tant que nouvelle entrée dans la table de routage, à l'aide du « contrôleur ».

❖ **Serveur à la fonction de passerelle**

Il permet principalement au serveur d'envoyer différents types de données à la passerelle correspondante. Dans notre cas, les données représentent les règles d'accès. Pour cette méthode, seules l'adresse IP de la passerelle et le port d'écoute sont nécessaires. Le contrôleur SDN recherchera alors cette passerelle dans la topologie en utilisant à nouveau l'hôtetracker, ce qui permet également de trouver le commutateur directement lié à la passerelle.

La fonction s'appelle "Envoi du package à la passerelle" et prend comme paramètres un objet JSON contenant :

- IP: adresse IP de la passerelle ;
- type : UDP ou TCP ;
- port : sur lequel la passerelle écoute ;
    - payload: les données à envoyer à la passerelle (String / base64 / json / etc.).

Ce module fonctionne comme suit :

1- Une commande est envoyée depuis le serveur avec l'adresse IP de la passerelle, le port, le type de protocole à utiliser pour communiquer avec la passerelle et la charge utile contenant les règles à appliquer à l'intérieur de la passerelle.
2- Le serveur JSON RPC, situé dans le contrôleur SDN, reçoit la demande et vérifie le format. Ensuite, extrait les paramètres de l'objet JSON.
3- L'hôte est ensuite appelé pour collecter les paramètres liés à la passerelle.
4- La création de l'en-tête Ethernet.
5- La création de l'en-tête IP.
6- La création de l'en-tête UDP.
7- La création de l'ensemble du paquet, composée d'un en-tête UDP avec la charge utile (les règles d'accès), encapsulée dans un paquet IP, à nouveau encapsulée dans un paquet Ethernet.
8- Ce paquet est ensuite transporté en utilisant le message « paquet sortant ».

9- Enfin, ce paquet est envoyé au commutateur correspondant (à l'aide du "DPID" déjà récupéré à l'aide de la piste hôte), puis remis à la passerelle.

5.3.5 Cas d'utilisation dans un Smart Space ou espace intelligent

Comme indiqué précédemment, les infrastructures de la prochaine génération, et en particulier celles liées à la santé et à l'environnement, introduiront un très grand nombre de dispositifs IoT. Notre approche est donc illustrée par un cas d'utilisation du domaine d'un smart space. Il s'agit de faire la gestion de l'automatisation de la couche d'accès et celle de transport des différentes Smart Spaces et où chaque entité peut être considérée comme une Smart Space. La Figure 5.12 représente les éléments clés de ce système.

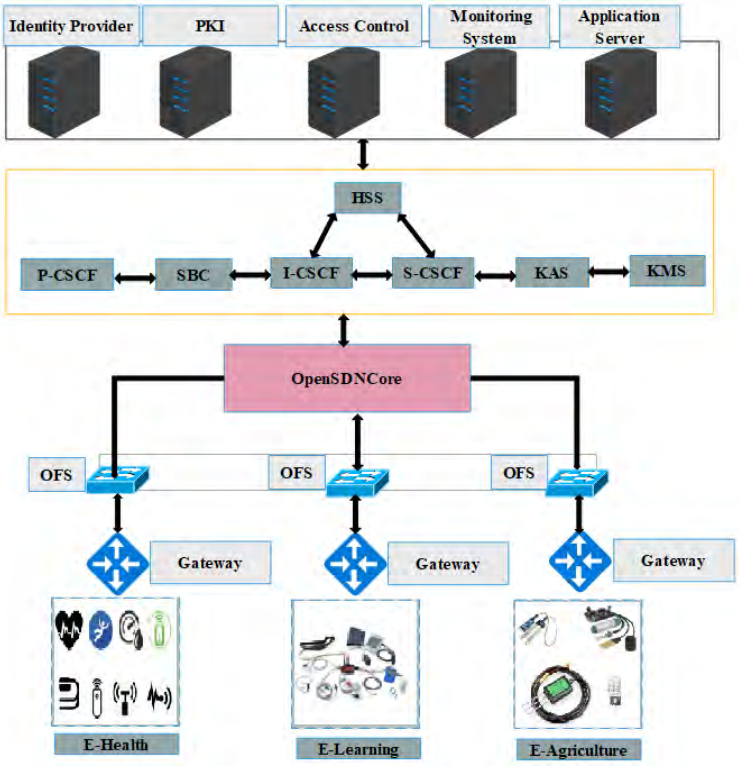

Figure 5.12: Architecture fonctionnelle proposée

L'architecture fonctionnelle de la solution proposée consiste en l'intégration de la technologie SDN dans une plate-forme IMS/KMS. Cette intégration permet de faciliter aux utilisateurs l'accès au réseau IMS/KMS mais aussi de diversifier les terminaux (Smartphone, tablette, ordinateur, etc.). L'architecture de la solution comprend quatre couche à savoir la couche d'accès, la couche transport, la couche contrôle et la couche application.

5.3.5.1 Couche accès

La couche accès est composée essentiellement du réseau d'accès, des terminaux, de la passerelle et des IoT devices. IMS fournit un cadre normalisé pour la création et le déploiement de services sur tous les réseaux IP, indépendamment de la technologie d'accès, prenant en charge les environnements de télécommunications mobiles et sans fil. Dans le dispositif IoT, on a une passerelle IoT pour interagir avec la plate-forme IMS/KMS. Cette passerelle est l'équipement chargé de récupérer les données recueillies par les IoT devices et de les transmettre au réseau d'accès afin que les terminaux puissent y accéder. La passerelle joue un rôle important dans l'intégration des technologies SDN et IoT. Elle peut prendre en charge plusieurs types de dispositifs à travers une architecture pilote [216]. Elle peut également être caractérisée par sa représentation des ressources dans différents formats pour faciliter son intégration à d'autres applications web [217].

5.3.5.2 Couche transport

La couche transport est constituée de la partie SDN. Elle est composée du NOS qui représente le système d'exploitation, des différents Switch, d'un contrôleur chargé de piloter les équipements. Le concept SDN découple le plan de contrôle et le plan de données en concentrant les décisions de transfert de données dans un contrôleur central (logiciel). Un protocole normalisé commun qui définit la communication entre le contrôle et le plan de transfert de données est Openflow. Dans l'architecture actuelle, la couche de transport correspond directement au plan d'acheminement de données du réseau SDN. Ce plan est constitué de commutateurs Openflow. Chaque commutateur Openflow contient une ou plusieurs tables de flux et une couche d'abstraction qui communique avec le contrôleur.

5.3.5.3 Couche contrôle

La couche contrôle est constituée globalement d'un condensé des entités IMS et KMS. La partie IMS est composée du P-CSCF, du SBC, du I-CSCF, du S-CSCF et du HSS. Dans la partie KMS nous avons les entités Kurento Media Server (KMS) et Kurento Application Server (KAS). Ces deux parties nous permettent de gérer les services IMS et WebRTC chacune en ce qui la concerne. Dans le scénario SDN, les décisions de transmission de données et les fonctions de contrôle de niveau supérieur sont exécutées par un contrôleur logiciel de type POX logiquement centralisé. La configuration du réseau est effectuée sur une abstraction réseau simplifiée et les configurations de flux de données sont communiquées aux dispositifs

d'acheminement de données via le protocole Openflow. Le plan de contrôle est implémenté à l'aide du système d'exploitation de réseau SDN (NOS) et d'un ensemble de modules logiciels qui exécutent les diverses fonctions de contrôle du réseau. Le NOS expose les APIs ouvertes (APIs nord) pour permettre le développement de diverses applications qui peuvent interagir et fournir une meilleure gestion du réseau. Dans un tel scénario, les éléments de base IMS/KMS peuvent être implémentés en tant que modules pour une communication avec le NOS via les APIs nord.

5.3.5.4 Couche application

La couche de contrôle IMS/KMS utilise plusieurs interfaces pour communiquer avec la couche de service qui offrent la capacité de fournir des applications implémentées par des serveurs d'applications (AS). Ces derniers communiquent avec le S-CSCF en utilisant le protocole SIP afin de réaliser le service du contrôle de session. Ils communiquent également avec le HSS via le protocole DIAMETER afin de récupérer ou de mettre à jour les informations de profil de l'abonné. Les AS sont le type le plus courant de serveurs d'applications utilisés dans les scénarios IMS. Ils prennent en charge les applications SIP et répondent aux requêtes avec des messages SIP afin de permettre le traitement ultérieur des requêtes utilisatrices au S-CSCF. Les applications SIP peuvent inclure le service de présence ou des sessions impliquant des éléments multimédias tels que la voix, la vidéo, le chat, les jeux, etc.

5.3.5.5 Résultats et discussion

Pour illustrer la fonctionnalité de l'architecture proposée et de l'automatisation de la couche contrôle et de la couche transport nous pouvons nous référer aux figures 5.13, 5.14 et 5.15 suivantes. Ces figures montrent l'ensemble des tests que nous avons effectués. Pour faire ces tests nous avons utilisé un simulateur mininet. Pour notre cas, mininet nous a permet d'émuler les différentes entités de la plateforme IMS/KMS grâce à l'automatisation des deux couches intermédiaires (contrôle et transport). Ainsi, les figures 5.13 et 5.14 montrent le lancement du programme via la commande python automatisation.py. Ce programme prend en charge l'automatisation de la couche contrôle et de la couche transport. L'exécution d'un tel programme se déroule comme suit : le terminal crée le réseau et ajoute le contrôleur dont le rôle est de piloter les différents équipements du réseau. Ensuite on assiste à la création des hosts, des Switch et des liens entre les équipements. C'est seulement après cette étape que le système procède au démarrage des équipements du réseau (contrôleur, commutateurs, CLI, etc.) juste à

la suite de la configuration des ordinateurs. Si le processus se déroule correctement le terminal affiche mininet.

La figure 5.13 met en relief les tests de communication de l'automatisation de la plateforme IMS/KMS. En faisant un test global par pingall on remarque un échange de flux entre les différentes entités.

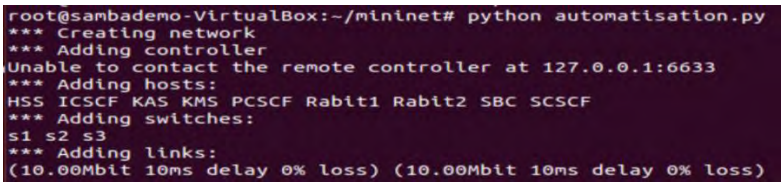

Figure 5.13: Première partie du lancement du programme

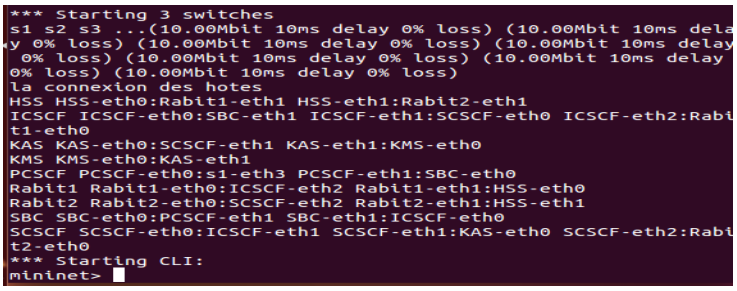

Figure 5.14: Lancement du programme suite et fin

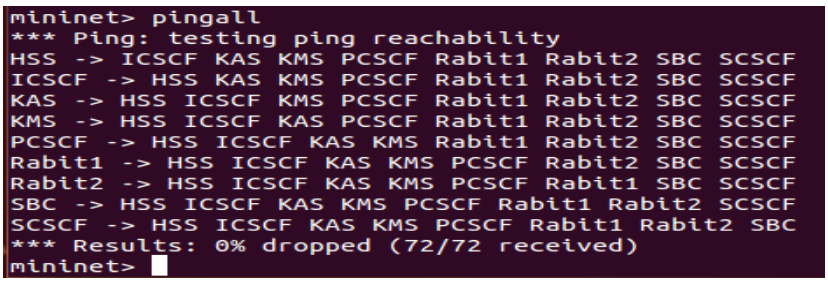

Figure 5.15: Test de communication de l'automatisation de la plateforme IMS/KMS

En nous fondant sur ces tests nous pouvons dire que cette proposition constitue un apport important pour la gestion des données recueillies par les IoT devices. En ce sens qu'elle offre beaucoup de facilité aux utilisateurs et diversifie les terminaux. Elle permet également d'automatiser la plate-forme IMS-KMS. Cette automatisation est plus qu'importante. En effet, lorsqu'un composant repose sur des processus manuels alors il peut devenir un goulot d'étranglement. Recourir à des outils d'administration et des commutateurs de Datacenter qui autorisent des niveaux d'automatisation élevés permet aux services informatiques d'éliminer ce type de goulots. Parallèlement, l'automatisation réseau contribue à accroître l'agilité et la

fiabilité, à accélérer les déploiements et à améliorer les performances des applications métier stratégiques.

5.4 CONCLUSION :

Dans ce chapitre, nous avons proposé un modèle de middleware IoT utilisant plusieurs espaces intelligents aussi bien pour la formation que pour le e-agriculture. Ensuite, pour faciliter l'administration de plusieurs espaces intelligents, nous avons fait appel au réseau programmé dit SDN, afin de faciliter d'une part l'accès et la gestion des utilisateurs et d'autre part d'automatiser les couches contrôle et transport.

En somme, le modèle proposé peut également être utilisé pour une gestion à plus grande échelle de plusieurs infrastructures intelligentes, par exemple dans une ville intelligente ou zone rurale intelligente. Il convient aussi de noter que notre approche a le mérite d'être une contribution aux efforts de faciliter l'accès aux services publics aussi bien pour la santé, l'agriculture et la formation. Elle apporte une solution concrète à résoudre les problèmes dans les secteurs fondamentaux de l'émergence d'un pays.

Conclusion Générale

Les travaux présentés dans cette thèse se concentrent principalement sur l'articulation entre quatre technologies différentes, WebRTC, IoT/WoT, IMS et SDN pour l'usage du middleware IoT afin de propose, d'une part un modèle d'espace intelligent aussi bien dans les villes que dans des zones rurales reculées prenant en charge des services de e-santé, e-agriculture et e-formation, et d'autre part une solution de gestion des réseaux complexes des objets connectés utilisés.

Nous avons amorcé les travaux de cette thèse par un état de l'art de l'IoT. C'est ce qui nous a permis de délimiter le champ de nos recherches. Nous nous sommes particulièrement intéressés aux QoS dans les middleware IoT. L'analyse des besoins en QoS des applications IoT dans les middlewares IoT donne à constater que WebRTC et le WoT sont des technologies adaptées non seulement à la collecte des données et à la visualisation mais aussi à la gestion de la communication audio/vidéo en temps réel utile dans nos domaines de prédilection que sont e-santé, e-formation et e-Agriculture.

Par la suite, l'étude menée sur les middleware IoT révèle, certes, des exigences de conception des plateformes orientées services, mais ne tient pas compte de l'intégration des technologies WebRTC, WoT et IMS dans le contexte d'un middleware IoT. Fort de ce qui précède, nous avons orienté nos recherches sur les normes, les exigences et les bonnes pratiques en matière de QoS en vue d'obtenir des éléments pertinents de conception d'une plateforme de mutualisation.

Par ailleurs, nous avons traité l'utilisation des middlewares IoT liés à e-santé, e-formation et e-Agriculture. Nous nous intéressons à des locuteurs dans un espace intelligent défini comme un environnement centré-utilisateur instrumenté par un ensemble de capteurs et d'actionneurs connectés. Nous avons analysé les capacités nécessaires pour permettre à un participant d'une session WebRTC d'impliquer dans cette même session, les flux induits par les objets connectés appartenant au SS d'un utilisateur quelconque de la session. L'accès à un tel environnement, peut être soit passif via l'observation d'informations contextuelles fournies par les capteurs, soit actif par l'exécution d'une commande d'un actionneur, soit une combinaison des deux. Cette approche recèle un gisement de nombreux nouveaux usages. Nous avons limité notre analyse à ceux concernant l'exercice distant d'une expertise et d'un savoir-faire.

À cet effet, d'un point de vue technique, il s'agit d'articuler de façon contrôlée les quatre technologies différentes que sont : WebRTC, l'IMS, l'Internet des objets (IoT)/Web des objets (WoT) et le SDN. Nous avons procédé dans un premier temps à une extension de WebRTC par WoT pour fournir à tout utilisateur d'une session WebRTC, un accès aux objets connectés de l'espace intelligent de tout autre participant à la session, en mettant l'accent sur la sécurisation de cet accès ainsi que sur sa conformité aux exigences de respect de la vie privée de l'utilisateur concerné. Le positionnement de notre approche dans le contexte des services de communication opérant dans les villes connectées, impose la prise en compte de multiples espaces intelligents et variés induisant chacun ses propres politiques de routage de qualité de service et de sécurité. Pour répondre à nos objectifs, il devient nécessaire au cours d'une session WebRTC, d'identifier, de sélectionner, de déployer et d'appliquer les règles de routage, QoS et de sécurité de façon à garantir un accès rapide aux différents espaces intelligents concernés. La seconde partie de notre travail consiste précisément à proposer une solution à cette problématique induite par des espaces intelligents distribués sur tout le réseau.

Par conséquent nous avons proposé dans cette thèse des contributions qui ont fait l'objet de 7 publications scientifiques deux IEEE (avec un Best Paper), quatre Springer et une IJACSA autour des technologies et standards tels que le WebRTC, WoT, IMS et SDN.

Notre travail s'est décliné en deux contributions principales.

En ce qui concerne la prise en charge des patients atteints de maladies chroniques dans les zones rurales et la télé-pédiatrie, dans les pays en voie de développement, nous avons proposé une plateforme utilisant des API REST, la technologie WebRTC et le concept de WoT pour mettre en place un dispositif de consultation à distance se rapprochant du présentiel. Cette solution pourra contribuer substantiellement à réduire la mortalité infanto-juvénile dans les pays en voie de développement surtout dans les zones rurales dépourvues de spécialistes en médecine. De la sécurisation native de WebRTC, nous avons dérivé un certain nombre de mécanismes permettant de sécuriser le lien entre la passerelle et le client WebRTC ainsi que le contrôle d'accès au espace intelligent.

Dans le cas de la télé-pédiatrie, le patient peut communiquer en temps réel avec le médecin en audio/vidéo via WebRTC, parallèlement il peut transférer au médecin toutes les informations collectées à partir des capteurs physiologiques dont il est muni. Le médecin pourra ainsi évaluer l'état de santé du patient en se basant à la fois sur les échanges réalisés durant la communication audio/vidéo et en même temps en analysant les données physiologiques reçues

en temps réel. La communication peut aussi se faire via des terminaux compatibles avec le protocole SIP.

Le deuxième cas d'usage est celui de la surveillance continue de l'état de santé d'un patient. Ce cas d'usage concerne principalement des personnes atteintes de maladies chroniques et vivant en zone rurale. En cas de détection d'anomalie, l'infirmier en charge du patient notifie par appel ou SMS au médecin généraliste ou spécialiste pour des conseils ou diagnostiques plus poussés.

La deuxième contribution propose l'utilisation des réseaux programmés dits SDN comme réseau de transport pour mieux prendre en charge la complexité de gestion et le nombre important des objets connectés. Le principe de notre approche consiste à centraliser les décisions concernant la gestion des différents espaces intelligents. Etant donné que les préoccupations en matière de routage sont intimement liées à celles de la sécurité, le contrôleur SDN apparait clairement comme un outil prometteur pour résoudre ces problèmes. Nous avons développé sur cette base une architecture originale permettant à un utilisateur de gérer dynamiquement et de manière efficace et simple l'accès à ses différents espaces intelligents. Cette approche est finalement illustrée dans le domaine du e-santé, du e-formation et du e-Agriculture en démontrant la possibilité de gérer une infrastructure d'une ville intelligente.

Perspectives

Les principaux défis auxquels le secteur de la santé actuel est confronté, sont principalement liés à la gestion des données, au manque d'infrastructure informatique, à la répartition déséquilibrée des ressources médicales et aux dispositifs médicaux liés à l'Internet des objets. Pour cette raison, les organisations de santé doivent adapter des techniques avancées et nouvelles pour relever ces défis. La nouvelle vision de la technologie 5G peut aider les fournisseurs de services de santé à améliorer ces processus, et les compagnies de téléphonies peuvent jouer un rôle dans cette chaîne de valeur. De plus, la tendance actuelle est d'exploiter les nouvelles capacités et performances réseau permises par la 5G au niveau de la couche d'application, et en particulier avec les applications Web telles que WebRTC et le monde IoT afin de prendre en charge beaucoup plus de périphériques avec une latence très faible.

De plus, le couplage de la 5G avec le SDN et le NFV semble être une solution très prometteuse pour le réseau de prochaine génération, en particulier pour les opérateurs de télécommunications.

Annexes

Les résultats obtenus dans cette thèse ont fait l'objet de 7 publications dont :

1. Deux publications IEEE

[1]. K. Gueye, N. Kag-Teube, S. Ouya and D. E. Moussavou, "Proposal for a universal access solution to care in rural areas : Case of Senegal," 2018 20th International Conference on Advanced Communication Technology (ICACT), Chuncheon-si Gangwon-do, Korea (South), 2018, pp. 1-1.

URL:
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8323865&isnumber=8323471

[2]. K. GUEYE, B. M. DEGBOE, S. OUYA and N. KAG-TEUBE, "Proposition of Health Care System Driven by IoT and KMS for Remote Monitoring of Patients in Rural Areas : Pediatric Case," 2019 21st International Conference on Advanced Communication Technology (ICACT), PyeongChang Kwangwoon_Do, Korea (South), 2019, pp. 676-680.

URL:
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8702009&isnumber=8701885

2. Quatre publications Springer

[3]. Gueye K., Boko U.H.S., Degboe B.M., Ouya S. (2020) Contribution to the Setting of an Online Platform on Practical Application for the Science, Technology, Engineering and Mathematics (STEM): The Case of Medical Field. In: Auer M., Tsiatsos T. (eds) The Challenges of the Digital Transformation in Education. ICL 2018. Advances in Intelligent Systems and Computing, vol 916. Springer, Cham.

https://link.springer.com/chapter/10.1007/978-3-030-11932-4_29

DOI :https://doi.org/10.1007/978-3-030-11932-4_29

[4]. Degboe B.M., Boko U.H.S., Gueye K., Ouya S. (2019) Contribution to the Setting up of a Remote Practical Work Platform for STEM : The Case of Agriculture. In: Mendy G., Ouya S., Dioum I., Thiaré O. (eds) e-Infrastructure and e-Services for Developing Countries. AFRICOMM 2018. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 275. Springer, Cham.

DOI :https://doi.org/10.1007/978-3-030-16042-5_9
https://link.springer.com/chapter/10.1007/978-3-030-16042-5_9#citeas

[5]. Ndiaye L., Gueye K., Ouya S., Mendy G. (2019) Contribution to Improving the Presence Base of VoIP Servers for Sending and Receiving Messages. In: Mendy G., Ouya S., Dioum I., Thiaré O. (eds) e-Infrastructure and e-Services for Developing Countries. AFRICOMM 2018. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 275. Springer, Cham.

DOI : https://doi.org/10.1007/978-3-030-16042-5_6
https://link.springer.com/chapter/10.1007/978-3-030-16042-5_6

[6]. Ngo Bilong J.R., Gueye K., Mendy G., Ouya S. (2019) Access Control Model Based on Dynamic Delegations and Privacy in a Health System of Connected Objects. In: Mendy G., Ouya S., Dioum I., Thiaré O. (eds) e-Infrastructure and e-Services for Developing Countries. AFRICOMM 2018. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 275. Springer, Cham

DOI : https://doi.org/10.1007/978-3-030-16042-5_11
https://link.springer.com/chapter/10.1007/978-3-030-16042-5_11

3. Une publication IJACSA

[7]. DIOUF, Samba, GUEYE, Keba, OUYA, Samuel, et al. Proposal for a Feature Automation Solution for an IMS-KMS-IoT Platform based on SDN. INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS, 2018, vol. 9, no 9, p. 500-505.

http://dx.doi.org/10.14569/IJACSA.2018.090963
DOI: 10.14569/IJACSA.2018.090963

Les travaux menés dans cette thèse on fait l'objet d'un Best paper à la 21ème conférence **IEEE/ ICACT2019 (Corée du sud)**

IEEE ComSoc
Technically Co-sponsored by the IEEE ComSoc

The 21st International Conference on
Advanced Communications Technology
(http://www.icact.org)
Feb. 17 ~ 20, 2019, Phoenix Park, Pyeongchang, Korea

icact 2019

Certificate of Outstanding Paper Award

1st Author: Mr. Kéba Gueye
Affiliation: LIRT/ESP/UCAD

Paper ID: 20190021
Title: Proposition of Health Care System Driven by IoT and KMS for Remote Monitoring of Patients in Rural Areas: Pediatric Case
Author(s): Kéba GUEYE *, Bessan M. DEGBOE *, Samuel OUYA*, Ngartabé KAG-TEUBE
Affiliation(s): *University Cheikh Anta Diop of Dakar, Senegal

This award is presented to Mr. Kéba Gueye for his/her excellent achievement in the 21st International Conference hosted by the Global IT Research Institute with IEEE Communication Society.

2019-02-18

Prof. Dae-Young Kim
ICACT2019 TPC Chair

Prof. Đỗ Năng Toàn, PhD
ICACT2019 General Chair

GIRI 1713 Obelisk, 216 Seohyunno, Bundang-gu, Sungnam Kyunggi-do, Korea(South) 13591 (eMail: office@icact.org)

Bibliographie

[1] S. Lucero et al., “Iot platforms: enabling the internet of things,” White paper,2016

[2] J. Mander, “Understanding the mobile-only internet user,” in Trends 17, the trends to watch in 2017, Globalwebindex, 2017.

[3] H. Mshali, T. Lemlouma, M. Moloney, and D. Magoni, “A survey on health monitoring systems for health smart homes,” International Journal of Industrial Ergonomics, vol. 66, pp. 26 – 56, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169814117300082

[4] M. Vega-Barbas, D. Casado-Mansilla, M. A. Valero, D. L. de Ipi˜na, J. Bravo, and F. Fl´orez, “Smart spaces and smart objects interoperability architecture (s3oia),”in 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, July 2012, pp. 725–730.

[5] L. Atzori, A. Iera, G. Morabito, “The Internet of Thingsμ A survey,” Computer Networks, vol 54, no. 15, pp. 2787-2805, October 2010.

[6] Ashton K., “That 'internet of things' thing, in the real world things matter more than ideas,” RFID Journal, 22 June 2009

[7] Gartner, http://www.gartner.com/it-glossary/internet-of-things, accès en Septembre 2016.

[8] I. Ishaq, D. Carels, G. K. Teklemariam, J. Hoebeke, F. V. d. Abeele, E. D. Poorter, I. Moerman, and P. Demeester, “Ietf standardization in the field of the internet of things (iot)μ a survey,” Journal of Sensor and Actuator Networks, vol. 2, no. 2, pp. 235–287, 2013.

[9] IERC, Internet of Things: From Research and Innovation to Market Deployment. IERC Cluster SRIA, 2014.

[10] R. Gerven, S. Jaarsma, and R. Wilhite, “Smart metering,” Kema, The Netherlands, July 2006.

[11] Europa, http://europa.eu.int/information_society/eeurope/ehealth, accès en Février 2017.

[12] ETSI TR 102 732 v1.1.1. “Machine-to-Machine communications (M2M); Use Cases of M2M applications for eHealth,” September 2013.

[13] Wu G., Talwar S., Johnsson K., Himayat N., Johnson K., “M2Mμ From Mobile to Embedded Internet,” IEEE Communications Magazine, 2011.

[14] Weyrich, M., Schmidt, J.-P., Ebert, C., “Machine-to-Machine Communication,” IEEE Software, August 2014.

[15] Info-communications Development Authority of Singapore (IDA), “Internet of things roadmap,” 2014.

[16] Ren Duan, Xiaojiang Chen, and Tianzhang Xing, “A qos architecture for iot,” International Conference on and 4th International Conference on Cyber, Physical and Social Computing, China, 2011.

[17] ETSI TS 102 690, “Machine-to-Machine communications (M2M); Functional architecture,” v2.1.1, October 2010.

[18] M.A. Razzaque, M. Milojevic-Jevric, S. Clarke, “Middleware for Internet of Thingsμ A Survey,” in Internet of Things Journal, IEEE, vol. 3, no. 1, February 2016.

[19] A. Al-Fuqaha, A. Khreishah, M. Guizani, A. Rayes, “Toward better horizontal integration among IoT services,” IEEE Communication Magazine, vol. 53, no. λ, pp. 72-79, Sep. 2015.

[20] I. Cha, Y. Shah, A. U. Schmidt, A. Leicher, and M. Meyerstein, “Security and trust for m2m communications 1,” 2009.

[21] H. Ning, H. Liu and L. Yang, “Cyber-entity security in the Internet of things,” Computer, vol. 46, no. 4, pp. 46-53, 2013.

[22] ETSI TS 118 109, “HTTP Protocol Binding,” v1.0.0, February 2015.

[23] Z. Shelby, ARM, K. Hartke, C. Bormann. “The Constrained Application Protocol (CoAP),” RFC 7252. Juin 2014.

[24] A. Banks and R. Gupta., “MQTT Version 3.1.1 Errata 01,” OASIS Approved Errata, 10 Décembre 2015.

[25] Internet Engineering Task Force, “XML Media Types,” RFC 7303, July 2014.

[26] Doug Crockford, “Google Tech Talksμ JavaScriptμ The Good Parts,” 7 February 2009.

[27] M. Starsinic, “System architecture challenges in the home M2M network,” Applications and Technology Conference (LlSAT), 2010 Long Island Systems, vol., no., pp.I,7, 7-7 May 2010.

[28] eCall project, https://ec.europa.eu/digital-single-market/en/news/ecall-all-new-cars-april-2018, accès en Janvier 2017.

[29] Booth, C., “Chapter 2 : IP Phones, Software VoIP, and Integrated and Mobile VoIP,” Library Technology Reports. 46 (5): 11–19, 2010.

[30] Church S. and Pizzi S., “Audio Over IP,” Focal Press., p. 191, 2010.

[31] I. Petiz, P. Salvador, and A. N. Nogueira. “Characterization and modeling of m2m video surveillance traffic,” In IARIA Int. Conf. on Advances in Future Internet - AFIN, August 2012.

[32] Saamer Akhshabi; Ali C. Begen; Constantine Dovrolis, “An Experimental Evaluation of Rate-Adaptation Algorithms in Adaptive Streaming over HTTP,” In Proceedings of the second annual ACM conference on Multimedia systems (MMSys '11). New York, NY, USA: ACM. 2011.

[33] N. Nikaein, M. Laner, K. Zhou, P. Svoboda, D. Drajic, M. Popovic, S. Krco, “Simple traffic modeling framework for machine type communication,” International Symposium on Wireless Communication Systems, pp. 1-5, Aug 2013.

[34] D. Katusic, P. Skocir, I. Bojic, M. Kusek,., Jezic, G., Desic, S., Huljenic, D., “Universal identification scheme in machine-to-machine systems,” Inμ 2013 12th International Conference on Telecommunications (ConTEL), pp. 71–78, 2013.

[35] IEEE, “IEEE standard glossary of software engineering terminology,” 1990.

[36] L. Skorin-Kapov and M. Matijasevic, “Analysis of qos requirements for e-Health services and mapping to evolved packet system qos classes,” International Journal of Telemedecine and Applications. July 2010.

[37] L. Wen-xiang, X. Jun, and J. Hao, “Queuing states analysis on a hybrid scheduling strategies for heterogeneous traffics in iot,” Computer Science & Service System (CSSS), pp.1007-1008, August 2012.

[38] J.R. Gallego, A. Hernandez-Solana, M. Canales, J. Lafuente, A. Valdovinos, J. Fernandez-Navajas, “Performance Analysis of Multiplexed Medical Data Transmission for Mobile Emergency Care Over the UMTS Channel,” IEEE Trans. Information Technology in Biomedicine, vol. 9, no. 1, pp. 13-22, Mar. 2005.

[39] M. Shaw and D. Garlan, “Software architectureμ perspectives on an emerging discipline,” Vol. 1, p.12. Englewood Cliffs: Prentice Hall, 1996.

[40] R. T. Fielding & R. N. Taylor, “Architectural styles and the design of network-based software architectures,” p. 151, Doctoral dissertation: University of California, Irvine. 2000.

[41] Sun Microsystems, Inc. “RPCμ Remote Procedure Call Protocol Specification, Version 2,” RFC 1057. Juin 1988.

[42] A. D. Birrell, and B. J. Nelson, “Implementing remote procedure calls,” ACM Transactions on Computer Systems, pp. 39-59, March 1984.

[43] Sun Microsystems, Inc. “NFSμ Network File System Protocol Specification”. RFC 1094, March 1989.

[44] L. Richardson and S. Ruby, “RESTful web services,” O’Reilly Media, Inc., 2008.

[45] IBM, “Object Request Brokers,” November 2013.

[46] CORBA OMG, disponible à http://www.corba.org, 2017.

[47] Scott McLean, James Naftel and Kim Williams. “Microsoft .NET Remoting,” Microsoft Press. 2002.

[48] Oracle, “Remote Method Invocation specification,” 2017.

[49] W. Emmerich, “Software engineering and middlewareμ a roadmap”, Conference on The future of Software engineering, pp. 117-129, 2000.

[50] Enterprise Integration Patterns, “Hub and Spoke [or] Zen and the Art of Message Broker Maintenance,” November 2003.

[51] Snyder, Bruce; Bosanac, Dejan; Davies, Rob, “ActiveMQ in Action,” (1st ed.), Manning Publications, p. 375, March 2010.

[52] RedHat Fuse, http://www.redhat.com/about/news/press-archive/2012/6/red-hat-to-acquirefusesource, accès en January 2017.

[53] “Launch of RabbitMQ Open Source Enterprise Messaging,” Press release. February 8, 2007.

[54] OASIS standard, “OASIS Advanced Message Queuing Protocol,” Version 1.0, October 2012.

[55] OMG, “Data-Distribution Service for Real-Time Systems,” Version 1.0, December 2004.

[56] OASIS standard, “SOA RM - Reference Model for Service Oriented Architecture 1.0,” October 2006.

[57] Lapeira, Raul. “ESB is an architectural style, a software product, or a group of software products?”. Artifact Consulting. 2002.

[58] R. T. Fielding & R. N. Taylor, “Architectural styles and the design of network-based software architectures,” p. 151, Doctoral dissertation : University of California, Irvine. 2000.

[59] oneM2M TS v1.6.1, “oneM2M functional architecture,” January 2015.

[60] Linux Foundation, “IoTivity Open Source Project Announces Preview Release,”14 January 2015.

[61] OCF, “IoT Standards Get a Big Pushμ Meet the Open Connectivity Foundation (OCF),” 23 February 2016.

[62] AllJoyn, AllSeen Alliance, accès en ligne à https://allseenalliance.org.

[63] Fiware, accès à https://www.fiware.org/

[64] IERC Cluster SRIA, “Internet of Thingsμ From Research and Innovation to Market Deployment,” 2014.

[65] Y. Banouar, S. Reddad, C. Diop, C. Chassot and A. Zyane, “Monitoring solution for autonomic Middleware-level QoS management within IoT systems,” 2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA), Marrakech, 2015, pp. 1-8.

[66] S.-Y. Yu, Z. Huang, C.-S. Shih, K.-J. Lin, J. Hsu, “Qos oriented sensor selection in iot system,” in : IEEE and Internet of Things (iThings/CPSCom), 2014.

[67] K.-J. Lin, N. Reijers, Y.-C. Wang, C.-S. Shih, and J. Y. Hsu, “Building Smart M2M Applications Using the WuKong Profile Framework,” 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, pp. 1175-1180, August 2013.

[68] W. Heinzelman, A. Murphy, H. Carvalho, M. Perillo, “Middleware to support sensor network applications,” Network, IEEE, vol. 18, issue 1, pp. 6-14, 2004.

[69] A. Sîrbu, S. Caminiti, P. Gravino, V. Loreto, V. Servedio, F. Tria, “A new platform for Human Computation and its application to the analysis of driving behaviour in response to traffic information,” CCS14 Proceedings in Human Computation, 2014.

[70] A. Banks and R. Gupta., “MQTT Version 3.1.1 Errata 01,” OASIS Approved Errata, 10 Décembre 2015.

[71] D. Guinard and V. Trifa, Building the Web of Things. Manning Publications Co, 2016.

[72] Andrew (PrismTech) Foster. Messaging Technologies for the Industrial Internet and the Internet of Things, 2014.

[73] C. Severance. Roy t. fielding: Understanding the rest style. Computer, 48(6):7–9, June 2015.

[74] Z. B. Babovic, J. Protic, and V. Milutinovic. Web performance evaluation for internet of things applications. IEEE Access, 4:6974–6992, 2016.

[75] W. Shang, Y. Yu, R. E. Droms, and L. Zhang. Challenges in iot networking via tcp/ip architecture. Number 2, page 7,2016.

[76] T. Dierks and E. Rescorla. The transport layer security (tls) protocol version 1.2. RFC 5246, RFC Editor, August 2008.

[77] K. Bhargavan, B. Blanchet, and N. Kobeissi. Verified models and reference implementations for the tls 1.3 standard candidate. In 2017 IEEE Symposium on Security and Privacy (SP), pages 483–502, May 2017.

[78] X. Li, J. Xu, Z. Zhang, D. Feng, and H. Hu. Multiple handshakes security of tls 1.3 candidates. In 2016 IEEE Symposium on Security and Privacy (SP), pages 486–505, May 2016.

[79] T. Savolainen, N. Javed, and B. Silverajan. Measuring energy consumption for restful interactions in 3gpp iot nodes. In 2014 7th IFIP Wireless and Mobile Networking Conference (WMNC), pages 1–8, May 2014.

[80] M. Belshe, R. Peon, and M. Thomson. Hypertext transfer protocol version 2 (http/2). RFC 7540, RFC Editor, May 2015.

[81] Daniel Stenberg. Http2 explained. SIGCOMM Comput. Commun. Rev., 44(3):120–128, July 2014.

[82] Z. Shelby, K. Hartke, and C. Bormann. The constrained application protocol (coap). RFC 7252, RFC Editor, June 2014.

[83] C. Bormann, S. Lemay, H. Tschofenig, K. Hartke, B. Silverajan, and B. Raymor. Coap (constrained application protocol) over tcp, tls, and websockets. RFC 8323, RFC Editor, February 2018.

[84] H. V. Nguyen and L. L. Iacono. Rest-ful coap message authentication. In 2015 InternationalWorkshop on Secure Internet of Things (SIoT), pages 35–43, Sept 2015.

[85] N. Correia, D. Sacramento, and G. SchAijtz. Dynamic aggregation and scheduling in coap/observe-based wireless sensor networks. IEEE Internet of Things Journal, 3(6):923–936, Dec 2016.

[86] A. Kerenen M. Koster and J. Jimenez. Publish-Subscribe Broker for the Constrained Application Protocol. Rfc, RFC Editor, July 2017.

[87] E. Rescorla and N. Modadugu. Datagram transport layer security version 1.2. RFC 6347, RFC Editor, January 2012.

[88] M. Panwar and A. Kumar. Security for iot: An effective dtls with public certificates. In 2015 International Conference on Advances in Computer Engineering and Applications, pages 163–166, March 2015.

[89] S. Raza, H. Shafagh, K. Hewage, R. Hummen, and T. Voigt. Lithe: Lightweight secure coap for the internet of things. IEEE Sensors Journal, 13(10):3711–3720, Oct 2013.

[90] J. Granjal, E. Monteiro, and J. SAą Silva. Security for the internet of things: A survey of existing protocols and open research issues. IEEE Communications Surveys Tutorials, 17(3):1294–1312, thirdquarter 2015.

[91] V. Lakkundi and K. Singh. Lightweight dtls implementation in coap-based internet of things. In 20th Annual International Conference on Advanced Computing and Communications (ADCOM), pages 7–11, Sept 2014.

[92] N. Tantitharanukul, K. Osathanunkul, K. Hantrakul, P. Pramokchon, and P. Khoenkaw. Mqtt-topics management system for sharing of open data. In 2017 International Conference on Digital Arts, Media and Technology (ICDAMT), pages 62–65, March 2017.

[93] Edited by Andrew Banks and Rahul Gupta. Mqtt version 3.1.1. Oasis standard, 29 October 2014.

[94] J. E. Luzuriaga, J. C. Cano, C. Calafate, P. Manzoni, M. Perez, and P. Boronat. Handling mobility in iot applications using the mqtt protocol. In 2015 Internet Technologies and Applications (ITA), pages 245–250, Sept 2015.

[95] A. Stanford-Clark and H. Linh Troung. MQTT For Sensor Networks (MQTT-SN) Protocol Specification Version 1.2. Mqtt.Org, 2013.

[96] K. Govindan and A. P. Azad. End-to-end service assurance in iot mqtt-sn. In 2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC), pages 290–296, Jan 2015.

[97] Y. Xu, V. Mahendran,W. Guo, and S. Radhakrishnan. Fairness in fog networks: Achieving fair throughput performance in mqtt-based iots. In 2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC), pages 191–196, Jan 2017.

[98] C. Lesjak, D. Hein, M. Hofmann, M. Maritsch, A. Aldrian, P. Priller, T. Ebner, T. Ruprechter, and G. Pregartner. Securing smart maintenance services: Hardware-security and tls for mqtt. In 2015 IEEE 13th International Conference on Industrial Informatics (INDIN), pages 1243–1250, July 2015.

[99] Object Management Group (OMG). OpenDDS Developer's Guide, 2017.

[100] A. Corsaro. The Data Distribution Service Tutorial. 05 2014.

[101] G. Farabaugh G. Pardo-Castellote and R.Warren. An introduction to dds and data-centric communications. Real-Time Innovations, (August), 2005.

[102] G. B. Baptista, F. Carvalho, S. Colcher, and M. Endler A Middleware for Data-centric and Dynamic Distributed Complex Event Processing for IoT Real-time Analytics in the Cloud. (2001).

[103] J. Yang, K. SandstrAűm, T. Nolte, and M. Behnam. 2012. Data Distribution Service for industrial automation. In Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies Factory Automation (ETFA 2012). 1–8. https://doi.org/10.1109/ETFA.2012.6489544.

[104] M. Hamilton H. Choi S. Rhee G. Subramanian Y. Dai E. Sin S. Sonck Thiebaut, G. Pardo-Castellote and A. Bose. 2002. Real-Time Publish Subscribe (RTPS) Wire Protocol Specification. RFC. RFC Editor.

[105] J. F. Inglés-Romero, A. Romero-Garcés, C. Vicente-Chicote, and J. Martínez. 2017. A Model-Driven Approach to Enable Adaptive QoS in DDS-Based Middleware. IEEE Transactions on Emerging Topics in Computational Intelligence 1, 3 (June 2017), 176–187. https://doi.org/10.1109/TETCI.2017.2669187.

[106] S. Pradhan, W. Emfinger, A. Dubey, W. R. Otte, D. Balasubramanian, A. Gokhale, G. Karsai, and A. Coglio. 2014. Establishing Secure Interactions across Distributed Applications in Satellite Clusters. In 2014 IEEE International Conference on Space Mission Challenges for Information Technology. 67–74. https://doi.org/10.1109/SMC-IT.2014.17.

[107] A. Al-Fuqaha, A. Khreishah, M. Guizani, A. Rayes, and M. Mohammadi. 2015. Toward better horizontal integration among IoT services. IEEE Communications Magazine 53, 9 (September 2015), 72–79. https://doi.org/10.1109/MCOM.2015.7263375.

[108] Object Management Group (OMG). 2015. Data Distribution Service (DDS) Version 1.4. (March 2015), 1–20.

[109] Object Management Group (OMG). 2017. OpenDDS Developer's Guide. (2017).

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

[111] J. L. Fernandes, I. C. Lopes, J. J. P. C. Rodrigues, and S. Ullah. 2013. Performance evaluation of RESTful web services and AMQP protocol. In 2013 Fifth International Conference on Ubiquitous and Future Networks (ICUFN). 810–815. https://doi.org/10.1109/ICUFN.2013.6614932.

[112] E. C. M. van der Linden, Jonas Wallgren, and Peter Jonsson. 2017. A latency comparison of IoT protocols in MES.

[113] P. Saint-Andre. 2004. Extensible Messaging and Presence Protocol (XMPP): Core. RFC 3920. RFC Editor.

[114] A. Hornsby and R. Walsh. 2010. From instant messaging to cloud computing, an XMPP review. In IEEE International Symposium on Consumer Electronics (ISCE 2010). 1–6. https://doi.org/10.1109/ISCE.2010.5523293.

[115] J. Ramirez and C. Pedraza. 2017. Performance analysis of communication protocols for Internet of things platforms. In 2017 IEEE Colombian Conference on Communications and Computing (COLCOM). 1–7. https://doi.org/10.1109/ ColComCon.2017.8088198

[116] M. T. Jones, "Meet the Extensible Messaging and Presence Protocol (XMPP)," DeveloperWorks, 2009.

[117] D. Conzon, T. Bolognesi, P. Brizzi, A. Lotito, R. Tomasi, and M. A. Spirito. 2012. The VIRTUS Middleware: An XMPP Based Architecture for Secure IoT Communications. In 2012 21st International Conference on Computer Communications and Networks (ICCCN). 1–6. https://doi.org/10.1109/ICCCN.2012.6289309.

[118] D. Schuster, P. Grubitzsch, D. Renzel, I. Koren, R. Klauck, and M. Kirsche. 2014. Global-Scale Federated Access to Smart Objects Using XMPP. In 2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom). 185–192. https://doi.org/10.1109/iThings.2014.35.

[119] X. Che and S. Maag. 2013. A Passive Testing Approach for Protocols in Internet of Things. In 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing. 678–684. https://doi.org/10.1109/GreenCom-iThings-CPSCom.2013.124.

[120] A. Hornsby and E. Bail. 2009. µXMPP: Lightweight implementation for low power operating system Contiki. In 2009 International Conference on Ultra Modern Telecommunications Workshops. 1–5. https://doi.org/10.1109/ICUMT.2009.5345594

[121] H. Wang, D. Xiong, P. Wang, and Y. Liu. 2017. A Lightweight XMPP Publish/Subscribe Scheme for Resource-Constrained IoT Devices. IEEE Access 5 (2017), 16393–16405. https://doi.org/10.1109/ACCESS.2017.2742020.

[122] Extensible Messaging and Presence Protocol (XMPP): Core, IETF: RFC 6120, 2011, https://tools.ietf.org/html/rfc6120.

[123] P.Waher and Y.Doi, XEP-0322: Efficient XML Interchange (EXI) Format, 2013.

[124] T. Kamiya and J. Schneider, Efficient XML Interchange (EXI) Format 1.0, World Wide Web Consortium Recommendation REC Exi-20110310, 2011.

[125] Rosenberg, J. AND Schulzrinne, H. AND Camarillo, G. AND Johnston, A. AND Peterson, J. AND Spartks, R. AND Handley, M. AND E. Schooler. SIP : Session Initiation Protocol, jun 2002. RFC 3261

[126] D. Thangavel, X. Ma, A. Valera, H.-X. Tan, and C. K.-Y. Tan, "Performance evaluation of MQTT and CoAP via a common middleware," in Proceedings of the 9th IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP '14), pp. 1–6, IEEE, Singapore, April 2014.

[127] N. De Caro, W. Colitti, K. Steenhaut, G. Mangino, and G. Reali. 2013. Comparison of two lightweight protocols for smartphone-based sensing. In 2013 IEEE 20th Symposium on Communications and Vehicular Technology in the Benelux (SCVT). 1–6. https://doi.org/10.1109/SCVT.2013.6735994.

[128] J. Hosek, P.Masek, D. Kovac, and F. Kropfl, "M2M gateway: the centerpiece of future home," in Proceedings of the 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT '14), pp. 190–197, St. Petersburg, Russia, October 2014.

[129] OUAKIL L, PUJOLLE G (2008) Téléphonie sur IP. In: Eyrolles (éd) 2e edn. EYROLLES, p 466.

[130] Ouya S, Dahirou Gueye A, Sy K, Niane MT, Lishou C (2015) Contribution to reducing the effects of geographical separation between actors of virtual universities: Proposal of an IP-SMSC integrating value-added services solutions. Proc 2015 Int Conf Interact Collab Learn ICL 2015 1145 1150. doi: 10.1109/ICL.2015.7318195.

[131] Ouya S (2015) ÉTUDE DE LA CONVERGENCE DES SERVICES DE TÉLÉCOMMUNICATIONS ET SES APPLICATIONS AUX ORGANISATIONS VIRTUELLES. Thèse: Télécommunications:Université Chiekh Anta Diop de Dakar.

[132] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., ... & Schooler, E. (2002). SIP : session initiation protocol.

[133] Crespi N, Beltran V (2013) WebRTC , the day after What ' s next for conversational services ? 46 52 Jennings C (2013) Real-time communications for the web. Commun … 279 284.

[134] Jennings C (2013) Real-time communications for the web. Commun … 279-284.

[135] Holmer S, Shemer M, Paniconi M (2013) Real-Time Communications in the Web. In: 2013 IEEE International Conference on Image Processing, ICIP 2013 - Proceedings. IEEE, p 1860 1864.

[136] Ouya S, Mendy G (2015) WebRTC platform proposition as a support to the educational system of universities in a limited Internet connection context. 47 52.

[137] Giuliana Carullo, Marco Tambasco, Mario Di Mauro, Maurizio Longo, A Performance Evaluation of WebRTC over LTE, 2016.

[138] Boni García, Luis López-Fernández, Micael Gallego, and Francisco Gortázar, Kurento : The Swiss Army Knife of WebRTC Media Servers, juin 2017.

[139] Carliss Young Baldwin,Kim B. Clar, Design Rules: The power of modularity, MIT Press, 2000.

[140] B. Bhuyan, H.K.D. Sarma, N. Sarma A survey on middleware for wireless sensor networks J. Wirel. Netw. Commun., 4 (2014), pp. 7-17

[141] J. Cecílio, P. Furtado Existing middleware solutions for wireless sensor networks Wireless Sensors in Heterogeneous Networked Systems, Computer Communications and Networks, Springer, Cham (2014), pp. 39-59, 10.1007/978-3-319-09280-5_4

[142] M.E.K. Ajana, H. Harroud, M. Boulmalf, M. Elkoutbi Middleware architecture in WSN Wireless Sensor and Mobile Ad-Hoc Networks, Springer, New York, NY (2015), pp. 69-94, 10.1007/978-1-4939-2468-4_4

[143] Bandyopadhyay, S., Sengupta, M., Maiti, S., Dutta, S.: Role of middleware for Internet of Things: a study. Int. J. Comput. Sci. Eng. Surv. 2(3), 94–105 (2011).

[144] Razzaque, M.A., Milojevic-Jevric, M., Palade, A., Clarke, S.: Middleware for Internet of Things: a survey. IEEE Internet of Things J. 3(1), 70–95 (2016).

[145] Radhika, J., Malarvizhi, S.: Middleware approaches for wireless sensor networks: an overview. Int. J. Comput. Sci. Issues (IJCSI) 9(3), 224–229 (2012).

[146] Raychoudhury, V., Cao, J., Kumar, M., Zhang, D.: Middleware for pervasive computing: a survey. Pervasive Mob. Comput. 9(2), 177–200 (2013)

[147] Alex, H., Kumar, M., Shirazi, B.: MidFusion: an adaptive middleware for information fusion in sensor network applications. Inf. Fusion 9(3), 332–343 (2008).

[148] Kim, J., Lee, J.W.: OpenIoT: an open service framework for the Internet of Things. In: 2014 IEEE World Forum on Internet of Things (WF-IoT). Institute of Electrical and Electronics Engineers (IEEE), March 2014. https://doi.org/10.1109/wf-iot.2014.6803126

[149] Heinzelman, W.B., Murphy, A.L., Carvalho, H.S., Perillo, M.A.: Middleware to support sensor network applications. IEEE Netw. 18(1), 6–14 (2004).

[150] Mamei, M., Zambonelli, F.: Field-Based Coordination for Pervasive Multiagent Systems. Springer, Heidelberg (2006). https://doi.org/10.1007/3-540-27969-5

[151] Liu, T., Martonosi, M.: Impala: a middleware system for managing autonomic, parallel sensor systems. In: ACM SIGPLAN Notices, vol. 38, pp. 107–118. ACM (2003).

[152] Kang, P., Borcea, C., Xu, G., Saxena, A., Kremer, U., Iftode, L.: Smart messages: a distributed computing platform for networks of embedded systems. Comput. J. 47(4), 475–494 (2004)

[153] Fok, C.L., Roman, G.C., Lu, C.: Agilla: a mobile agent middleware for self-adaptive wireless sensor networks. ACM Trans. Auton. Adapt. Syst. (TAAS) 4(3), 16 (2009).

[154] Hong, K., et al.: TinyVM: an energy-efficient execution infrastructure for sensor networks. Softw.: Pract. Exp. 42(10), 1193–1209 (2012)

[155] Marques, I.L., Ronan, J., Rosa, N.S.: TinyReef: a register-based virtual machine for wireless sensor networks. In: Sensors, pp. 1423–1426. IEEE (2009).

[156] Mottola, L., Murphy, A.L., Picco, G.P.: Pervasive games in a mote-enabled virtual world using tuple space middleware. In: Proceedings of 5th ACM SIGCOMM Workshop on Network and System Support for Games, p. 29. ACM (2006).

[157] Curino, C., Giani, M., Giorgetta, M., Giusti, A., Murphy, A.L., Picco, G.P.: TinyLIME: bridging mobile and sensor networks through middleware. In: Third IEEE International Conference on Pervasive Computing and Communications, PerCom 2005, pp. 61–72. IEEE (2005)

[158] Costa, P., Mottola, L., Murphy, A.L., Picco, G.P.: TeenyLIME: transiently shared tuple space middleware for wireless sensor networks. In: Proceedings of the International Workshop on Middleware for Sensor Networks, pp. 43–48. ACM (2006).

[159] Aberer, K., Hauswirth, M., Salehi, A.: A middleware for fast and flexible sensor network deployment. In: Proceedings of the 32nd International Conference on Very Large Data Bases, pp. 1199–1202. VLDB Endowment (2006).

[160] Kim, J., Lee, J.W.: OpenIoT: an open service framework for the Internet of Things. In: 2014 IEEE World Forum on Internet of Things (WF-IoT). Institute of Electrical and Electronics Engineers (IEEE), March 2014. https://doi.org/10.1109/wf-iot.2014.6803126.

[161] Guinard, D., Trifa, V., Mattern, F., Wilde, E.: From the Internet of Things to the web of things: resource-oriented architecture and best practices. In: Uckelmann, D., Harrison, M., Michahelles, F. (eds.) Architecting the Internet of Things, pp. 97–129. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19157-2_5

[162] Delicato, FC, Pires, PF, Batista, T.: Solutions middleware pour l'Internet des objets. Springer, Heidelberg (2013). https://doi.org/10.1007/978-1-4471-5481-5

[163] Evensen, P., Meling, H.: Sensewrap: a service oriented middleware with sensor virtualization and self-configuration. In: 2009 5th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), pp. 261–266. IEEE.

[164] Avilés-López, E., García-Macías, J.A.: TinySOA: a service-oriented architecture for wireless sensor networks. Serv. Oriented Comput. Appl. 3(2), 99–108 (2009).

[165] Alex, H., Kumar, M., Shirazi, B.: MidFusion: an adaptive middleware for information fusion in sensor network applications. Inf. Fusion 9(3), 332–343 (2008).

[166] Perera, C., Jayaraman, P.P., Zaslavsky, A., Georgakopoulos, D., Christen, P.: Mosden: an Internet of Things middleware for resource constrained mobile devices. In: 2014 47th Hawaii International Conference on System Sciences (HICSS), pp. 1053–1062. IEEE (2014).

[167] Benatallah, B., Motahari Nezhad, H.R.: Service oriented architecture: overview and directions. In: Börger, E., Cisternino, A. (eds.) Advances in Software Engineering. LNCS, vol. 5316, pp. 116–130. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89762-0_4

[168] Delamer, I.M., Lastra, J.L.M.: Service-oriented architecture for distributed publish/subscribe middleware in electronics production. IEEE Trans. Ind. Informat. 2(4), 281–294 (2006).

[169] Miluzzo, E., et al.: Sensing meets mobile social networks: the design, implementation and evaluation of the CenceME application. In: Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems, pp. 337–350. ACM (2008).

[170] Brouwers, N., Langendoen, K.: Pogo, a middleware for mobile phone sensing. In: Narasimhan, P., Triantafillou, P. (eds.) Middleware 2012. LNCS, vol. 7662, pp. 21–40. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35170-9_2

[171] J. O. Kephart, D. M. Chess, "The Vision of Autonomic Computing," IEEE Comp, 36 (1), Jan 2003.

[172] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An architecture for differentiated services," RFC 2475, IETF, December 1998.

[173] L. Skorin-Kapov and M. Matijasevic, "Analysis of qos requirements for e-Health services and mapping to evolved packet system qos classes," International Journal of Telemedecine and Applications. July 2010.

[174] T. H. Luan, L. Gao, Z. Li, Y. Xiang, and L. Sun, “Fog computing: Focusing on mobile users at the edge,” CoRR, vol. abs/1502.01815, 2015. [Online]. Available: http://arxiv.org/abs/1502.01815.

[175] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” in Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, ser. MCC ’12. New York, NY, USA: ACM, 2012, pp. 13–16. [Online]. Available: http://doi.acm.org/10.1145/2342509.2342513.

[176] P. Verma and S. K. Sood, “Fog assisted-iot enabled patient health monitoring in smart homes,” IEEE Internet of Things Journal, vol. 5, no. 3, pp. 1789–1796, June 2018.

[177] B. Garcia, L. Lopez-Fernandez, M. Gallego and F. Gortazar, "Kurento: The Swiss Army Knife of WebRTC Media Servers," in IEEE Communications Standards Magazine, vol. 1, no. 2, pp. 44-51, 2017.

[178] Zerkouk, M.: Modèles de contrôle d’accès dynamiques (Doctoral dissertation, University of sciences and Technology in Oran) (2015).

[179] A.A. El Kalam et al.: Or-BAC: un modèle de contrôle d’accès basé sur les organisations. Cahiers francophones de la recherche en sécurité de l’information 1, 30–43 (2003).

[180] O. Bettaz. N. Boustia, A. Mokhtari ‘’Dynamic delegation based on temporal context. Procedia Comput''. Sci. 96, 245–254 (2016)

[181] M. Ennahbaoui ‘’Contributions aux contrôles d’accès dans la sécurité des systèmes d’information’’ (2016).

[182] M.A. Abakar: ‘’Etude et mise en oeuvre d’une architecture pour l’authentification et la gestion de documents numériques certifiés : application dans le contexte des services en ligne pour le grand public’’ (Doctoral dissertation, Saint Etienne) (2012).

[183] A.A El Kalam., Y. Deswarte: Security model for health care computing and communication systems. In: Gritzalis, D., De Capitani di Vimercati, S., Samarati, P., Katsikas, S. (eds.) SEC 2003. ITIFIP, vol. 122, pp. 277–288. Springer, Boston, MA (2003). https://doi.org/10.1007/978-0-387-35691-4_24

[184] L. Zhang, G.J. Ahn, B.T Chu: A rule-based framework for role-based delegation and revocation. ACM Trans. Inf. Syst. Secur. (TISSEC) 6(3), 404–441 (2003).

[185] S. Chakraborty, I. Ray,: TrustBAC: integrating trust relationships into the RBAC model for access control in open systems. In: Proceedings of the Eleventh ACM Symposium on Access Control Models and Technologies, pp. 49–58. ACM, June 2006.

[186] A. Miege: Definition of a formal framework for specifying security policies. The Or-BAC model and extensions (Doctoral dissertation, Télécom ParisTech) (2005).

[187] E. Barka, R. Sandhu: A role-based delegation model and some extensions. In: Proceedings of the 23rd National Information Systems Security Conference, vol. 4, pp. 49–58, December 2000.

[188] D. F. Ferraiolo and D. R. Kuhn, “Role-based access control,” 1992, pp. 554 – 563.

[189] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, “Role-based access control models,” Computer, vol. 29, no. 2, pp. 38–47, Feb 1996.

[190] I. Ray, D. Mulamba, K.J. Han: A model for trust-based access control and delegation in mobile clouds. In: Wang, L., Shafiq, B. (eds.) DBSec 2013. LNCS, vol. 7964, pp. 242–257. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39256-6_16

[191] M.B. Ghorbel-Talbi, F. Cuppens, N. Cuppens-Boulahia, A. Bouhoula: Managing delegation in access control models. In: International Conference on Advanced Computing and Communications. ADCOM 2007, pp. 744–751. IEEE, December 2007.

[192] COOKING HACKS: e-Health Sensor Platform V2.0 for Arduino and Raspberry Pi [Biometric / Medical Applications], [online], 2015, Available: https://www.cookinghacks. com/documentation/tutorials/ehealth-biometric-sensor-platformarduino- raspberry-pi-medical.

[193] Eben Upton, Gareth Halfacree, ‘’Raspberry Pi User Guide’’, Wiley, 2014.

[194] S. Mohsin Reza, Md. Mahfujur Rahman, Md. Hasnat Parvez, Shamim Al Mamun, M. Shamim Kaiser, Innovative Approach in Web Application Effort & Cost Estimation using Functional Measurement Type. International Conference on Electrical Engineering and Information Communication Technology (ICEEICT), May 2015.

[195] F. Abtahi, B. Aslamy, I. Boujabir1, F. Seoane, and K. Lindecrantz,An Affordable ECG and Respiration Monitoring System Based on Raspberry PI and ADAS1000: First Step towards Homecare Applications. 16th Nordic-Baltic Conference on Biomedical Engineering,IFMBE Proceedings 48,DOI: 10.1007/978-3-319-12967-92.

[196] K. Siau,Health Care Informatics. IEEE Transactions on Information Technology in Biomedicine, vol. 7,March,2003.

[197] C. Octavian Truică, A. Boicea, Ionut Trifan, “CRUD Operations in Mon-goDB,” International Conference on Advanced Computer Science and Electronics Infor-mation, pp. 347-348, 2013.

[198] W.W. Smari, P. Clemente, Lalande, J.F. An extended attribute based access control model with trust and privacy: application to a collaborative crisis management system. Future Gener. Comput. Syst. 31, 147–168 (2014).

[199] P.P. Simanjuntak, P.T. Napitupulu, Silalahi, S.P., Kisno, P.N., Valešová, L. E-precision agriculture for small scale cash crops in Tobasa regency. In: 1st Nommensen International Conference on Technology and Engineering, Medan, Indonesia, 11–12 July 2017.

[200] I. Mohanraj, V. Gokul, R. Ezhilarasie, A Umamakeswari.: Intelligent drip irrigation and fertigation using wireless sensor networks. In: 2017 IEEE Technological Innovations in ICT for Agriculture and Rural Development (TIAR), Chennai, 2017, pp. 36–41 (2017).

[201] A.A. Garba: Smart water-sharing methods for farms in semi-arid regions. In: 2017 IEEE Technological Innovations in ICT for Agriculture and Rural Development (TIAR), Chennai, 2017, pp. 1–7 (2017).

[202] C. Yoon, M. Huh, S.-G. Kang, Park, J., Lee, C.: Implement smart farm with IoT technology. In: International Conference on Advanced Communications Technology (ICACT), Elysian Gangchon, Chuncheon Korea (South), February 11–14 2018 (2018).

[203] Vijayarajan, V., Krishnamoorthy, A., Abdul Gaffar, H., Deepika, R.: A novel approach to practices agriculture as e-farming service. Int. J. Innovation Sci. Res. 7(01), 1131–1134 (2018)

[204] S. Jaiganesh, K. Gunaseelan, V. Ellappan: IoT agriculture to improve food and farming technology. In: 2017 Conference on Emerging Devices and Smart Systems (ICEDSS), Tiruchengode, pp. 260–266 (2017)

[205] M. Sreeja, M. Sreeram: Teacher less classroom: a new perspective for making social empowerment a reality. In: 2017 IEEE Technological Innovations in ICT for Agriculture and Rural Development (TIAR), Chennai, 2017, pp. 186–193 (2017).

[206] B. Jose, S. Abraham: Exploring the merits of NoSQL: a study based on mongodb. In: 2017 International Conference on Networks & Advances in Computational Technologies (NetACT), Thiruvanthapuram, 2017, pp. 266–271 (2017).

[207] N. Feamster, J. Rexford, and E. Zegura, "The road to sdn: An intellectual history of programmable networks," SIGCOMM Comput. Commun. Rev., vol. 44, no. 2, pp. 87–98, Apr. 2014. [Online]. Available: http://doi.acm.org/10.1145/2602204.2602219

[208] M. Boussard, D. T. Bui, L. Ciavaglia, R. Douville, M. L. Pallec, N. L. Sauze, L. Noirie, S. Papillon, P. Peloso, and F. Santoro, "Software-defined lans for interconnected smart environment," in 2015 27th International Teletraffic Congress, Sept 2015, pp. 219–227.

[209] Q. Xiaofeng, L. Wenmao, G. Teng, H. Xinxin, W. Xutao, and C. Pengcheng, "Wot/sdn : web of things architecture using sdn,"China Communications, vol. 12, no. 11, pp. 1–11, November 2015.

[210] O. Flauzac, C. Gonz´alez, A. Hachani, and F. Nolot, "Sdn based architecture for iot and improvement of the security,"March 2015, pp. 688–693.

[211] P. Bull, R. Austin, E. Popov, M. Sharma, and R. Watson, "Flow based security for iot devices using an sdn gateway," in 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), Aug 2016, pp. 157–163.

[212] C. Vallati, E. Mingozzi, G. Tanganelli, N. Buonaccorsi, N. Valdambrini, N. Zonidis, B. Martnez, A. Mamelli, D. Sommacampagna, B Anggorojati, S. Kyriazakos, N. Prasad, F. Nieto, and O. Rodriguez, "Betaas: A platform for development and execution of machine-tomachine applications in the internet of things," WPC, pp. 1–21, 2015.

[213] I. Stojmenovic and S. Wen, "The Fog computing paradigm: Scenarios and security issues," Federated Conference on Computer Science and Information Systems (FedCSIS), 2014, pp. 1–8.

[214] F. F. C. C. NGNI, "OpenSDNCore – Research and Testbed for the carrier-grade NFV/SDN environment." [Online]. Available: https://www.opensdncore.org/

[215] E. W. Dijkstra, "A note on two problems in connexion with graphs,"Numerische mathematik, vol. 1, no. 1, pp. 269–271, 1959.

[216] C. Tranoris, S. Denazis, N. Mouratidis, P. Dowling, and J. Tynan, “Integrating OpenFlow in IMS Networks and Enabling for Future Internet Researchand Experimentation,” in The Future Internet, A. Galis and A. Gavras, Eds. Springer Berlin Heidelberg, 2013, pp. 77–88.

[217] Imène ELLOUMI, 2012. “Management of mobility between access networks and Quality of Service in an NGN / IMS approach. ” Thèse doctorat : University of Carthage (Tunisie)