République Algérienne Démocratique et Populaire
Université Abderrahmane MIRA de Béjaia
Faculté des Sciences Exactes

Département de Recherche Opérationnelle

Mémoire présenté pour l'obtention du diplôme de Master en Mathématiques Appliquées

Spécialité : Optimisation et Fiabilité des Réseaux de Communication

Proposition d'un protocole de routage hybride pour l'internet des objets

Présenté par :

LYDIA CHEKAL

Sous la direction de : Mme Z. AOUDIA
Et de : Dr N. ELSAKAAN

Défendu le 29/06/2025, devant le jury composé de :

M^{r} Khimoum Noureddine	M.C. Classe A	Président du jury	UAMB - Béjaïa.
M^{r} Aissani Djamil	Professeur	Examinateur	UAMB - Béjaïa.
M^{r} Sahli Ramzi	Doctorant	Examinateur	UAMB - Béjaïa.

Année Universitaire 2024–2025

Remerciements

Avant tout, je remercie Allah, le Tout-Puissant, de m'avoir accordé la force, la patience et la persévérance nécessaires pour mener à bien ce travail jusqu'à son aboutissement.

Je tiens à exprimer ma profonde reconnaissance à ma directrice de recherche, **Madame Zohra Aoudia**, pour son encadrement précieux, sa disponibilité constante et ses conseils éclairés. Son accompagnement tout au long de ce projet a été d'un soutien inestimable, tant sur le plan scientifique que personnel.

J'adresse également mes remerciements les plus sincères au **Dr. Nadim Elsakaan**, mon co-encadrant, pour son implication remarquable, sa rigueur et la richesse de ses apports. Grâce à son expertise, ses conseils méthodologiques et son suivi attentif, j'ai pu surmonter de nombreuses difficultés et progresser avec confiance. Sa disponibilité, sa pédagogie et sa bienveillance ont profondément marqué mon parcours.

Je tiens aussi à remercier chaleureusement **mademoiselle Zebida Chibane**, doctorante, pour son soutien constant, sa générosité et ses précieux conseils. Elle a su m'accompagner avec patience et encouragement, en m'apportant des explications claires et des suggestions pertinentes à chaque étape du travail. Sa présence à mes côtés a été d'un grand réconfort.

Je suis également reconnaissante envers les membres du jury, pour l'honneur qu'ils me font en acceptant d'évaluer ce travail, ainsi que pour leurs remarques et critiques constructives qui ne manqueront pas d'enrichir ma réflexion.

Je remercie par ailleurs l'ensemble des enseignants de l'Université pour la qualité de l'enseignement qu'ils m'ont transmis tout au long de mon cursus, et qui a constitué la base solide sur laquelle ce mémoire a été construit.

Enfin, j'exprime toute ma gratitude à toutes les personnes qui m'ont soutenue, de près ou de loin, tout au long de cette aventure, tant sur le plan académique que personnel.
Merci à toutes et à tous.

Dédicace

Je dédie ce travail, avec tout mon amour et ma gratitude, aux êtres les plus chers à mon cœur.

À ma mère, source infinie d'amour, de patience et de sacrifices. Merci pour tes prières silencieuses, ton soutien indéfectible et ta tendresse qui m'ont toujours portée dans les moments les plus difficiles. Tu es ma force et mon repère.

À mon père, pour sa confiance, sa sagesse, et ses encouragements constants. Merci pour ta présence rassurante, ton travail acharné et tes conseils précieux qui m'ont guidée tout au long de mon parcours.

À mon petit et unique frère, qui occupe une place très spéciale dans mon cœur. Merci pour ta joie de vivre, ton énergie contagieuse et ton affection sincère. Ta simple présence a souvent été un rayon de lumière dans mes journées chargées.

À ma famille toute entière, pour son amour inconditionnel, son soutien et ses encouragements tout au long de cette aventure.

À mes amis, pour leur écoute, leur patience et leurs mots bienveillants qui m'ont souvent redonné courage.

À mes collègues, de près comme de loin, pour les échanges enrichissants, le soutien mutuel et les souvenirs partagés tout au long de ce projet.

Ce mémoire est le fruit de tous ces liens précieux qui m'entourent.

Merci à vous, de tout cœur.

Lydia

Table des matières

Table des figures

Liste des Algorithmes

Liste des tableaux

Listes des abréviations

AODV	Ad hoc On-Demand Distance Vector
AMQP	Advanced Message Queuing Protocol
BLE	Bluetooth Low Energy
CoAP	Constrained Application Protocol
DAO	Destination Advertisement Object
DIO	DODAG Information Object
DIS	DODAG Information Solicitation
DODAG	Destination-Oriented Directed Acyclic Graph
GPU	Graphics Processing Unit
IGP	Interior Gateway Protocol
IoT	Internet of Things
LAN	Local Area Network
MQTT	Message Queuing Telemetry Transport
NFC	Near Field Communication
OSPF	Open Shortest Path First
QoS	Quality of Service
RERR	Route Error
RIP	Routing Information Protocol
RPL	Routing Protocol for Low-Power and Lossy Networks
RREP	Route Reply
RREQ	Route Request
RFID	Radio Frequency Identification
TPU	Tensor Processing Unit
UDP	User Datagram Protocol
XMPP	Extensible Messaging and Presence Protocol

Introduction générale

L'Internet des Objets (IoT) représente une révolution technologique majeure, visant à connecter des objets physiques au réseau Internet afin de collecter, échanger et exploiter des données en temps réel [6]. Cette interconnexion d'objets intelligents ouvre la voie à une large gamme d'applications dans des domaines aussi variés que la santé, la domotique, les transports, l'agriculture intelligente ou encore les villes intelligentes. L'IoT promet ainsi une amélioration significative de la qualité de vie, de l'efficacité énergétique, et de la prise de décision grâce à l'automatisation et à l'intelligence ambiante [6].

Cependant, malgré son potentiel considérable, l'IoT fait face à une série de défis techniques et scientifiques. Parmi les problématiques majeures figurent l'interopérabilité entre dispositifs hétérogènes, la gestion de l'énergie dans des environnements contraints, la sécurisation des données échangées, la robustesse face à la mobilité des nœuds et aux défaillances, ainsi que l'évolutivité dans des réseaux à grande échelle. Dans ce contexte, l'efficacité de la communication entre les objets connectés devient un enjeu central.

L'un des aspects fondamentaux de cette communication réside dans le routage, c'est-à-dire le processus permettant à un message d'atteindre efficacement sa destination à travers le réseau. Contrairement aux réseaux traditionnels, les réseaux IoT sont souvent formés de nœuds aux capacités limitées (en énergie, en mémoire, en calcul), disposés de manière dynamique, et sujets à des contraintes spécifiques telles que la densité variable, la perte de connectivité ou l'évolution fréquente de la topologie. Ces caractéristiques exigent le développement de stratégies de routage optimisées et adaptatives, capables de garantir la transmission des données avec un minimum de consommation de ressources tout en assurant la fiabilité du réseau.

C'est dans ce cadre que s'inscrit notre travail, qui vise à étudier, analyser et comparer différentes approches de routage dans les réseaux IoT, en particulier en lien avec la gestion des contraintes dynamiques. Nous portons une attention particulière à l'intégration de protocoles bien connus tels que RPL (Routing Protocol for Low Power and Lossy Networks) et AODV (Ad hoc On-Demand Distance Vector), et à leur performance dans des scénarios IoT réalistes.

Notre objectif est double : d'une part, proposer une approche hybride combinant détection communautaire et routage optimisé, et d'autre part, évaluer de manière comparative les performances de ces protocoles en termes de consommation d'énergie, et de robustesse face aux changements dynamiques du réseau.

Notre mémoire s'articule autour de quatre chapitres :

Le chapitre 1, intitulé « Concepts de base de l'internet des objets », pose les bases essentielles de notre étude. Il présente les principes clés de l'IoT, explore ses principaux domaines d'application, détaille les architectures typiques des systèmes IoT, et met en lumière les enjeux technologiques, économiques et sécuritaires auxquels ce domaine est confronté.

Le chapitre 2, intitulé « Protocoles de routage pour l'IoT », explore les principes fondamentaux du routage, ses différentes étapes, ainsi que les particularités du routage dans les réseaux classiques et dans l'IoT. Il présente en détail les protocoles RPL et AODV, en analysant leurs mécanismes, leurs avantages, leurs limites, et propose une comparaison approfondie entre ces deux approches dans le contexte spécifique de l'Internet des Objets.

Le chapitre 3, intitulé «Algorithme de routage hybride», constitue le cœur de notre contribution. Il présente de manière détaillée l'approche que nous avons développée, qui consiste à intégrer un algorithme de détection de communautés (Louvain), avec des protocoles de routage pour améliorer l'efficacité du réseau IoT. Chaque étape de notre méthode est décrite de façon rigoureuse, depuis la détection des communautés jusqu'à l'application des stratégies de routage intra et intercommunautaire, illustrant clairement notre démarche d'optimisation.

Enfin, le chapitre 4, « Évaluations et discussion des résultats », est dédié à l'analyse des performances de notre approche à travers une série de simulations expérimentales. Dans ce chapitre, nous présentons également le langage de programmation python utilisé pour implémenter et simuler notre solution, en mettant en avant ses avantages pour la modélisation et l'expérimentation dans un contexte IoT. Une évaluation comparative approfondie est réalisée entre notre approche et les protocoles de routage standards, permettant de mettre en évidence les gains apportés, les éventuelles limites rencontrées, ainsi que les perspectives d'amélioration pour des travaux futurs.

À travers ce mémoire, nous espérons apporter une contribution significative à la compréhension et à l'amélioration des mécanismes de routage dans l'Internet des Objets, en tenant compte des contraintes dynamiques et des besoins spécifiques de ce domaine en pleine expansion.

1

Concepts de base de l'internet des objets

Sommaire

Introduction

L'Internet a métamorphosé les interactions en reliant individus, machines et objets matériels, ce qui a conduit à la création de l'Internet des Objets et a ouvert de nouvelles avenues dans le secteur de la révolution digitale. Ce premier chapitre est dédié à la présentation des principes de base de l'IoT.

Nous entamerons ce chapitre par une étude détaillée de l'Internet des Objets, en établissant la définition de ce terme. Nous passerons aussi en revue ses multiples champs d'application et nous pencherons sur les différentes strates qui le constituent, sans oublier les protocoles et technologies de communication, pour finalement conclure sur les challenges liés à son évolution.

1.1 Historique de l'Internet des objets

Dans cette partie, nous mentionnons les événements clés qui ont jalonné la réalisation de l'IoT.

L'idée d'un réseau d'appareils intelligents a été introduite pour la première fois en 1982, avec le premier appareil relié à Internet à l'Université Carnegie Melon qui pouvait indiquer à son inventaire si les boissons récemment chargées étaient correctement refroidies [24].
En 1991, Mark Weiser a dévoilé l'idée de l'informatique omniprésente dans son article intitulé "L'ordinateur du XXI^{e} siècle", anticipant ainsi la vision moderne de l'Internet des objets.
Puis, en 1998 l'informatique omniprésente a suscité de l'intérêt en raison de sa capacité à intégrer de manière souple et efficace la technologie informatique dans le quotidien [24].
Ensuite, en l'an 2000, la compagnie LG dévoile son tout premier réfrigérateur intelligent relié à Internet. En outre, la technologie RFID (Identification par radiofréquence), qui constitue l'une des technologies fondamentales de l'IoT, a commencé à être largement mise en œuvre aux alentours de 2003 et 2004.
Par ailleurs, une démarche fort intéressante a été entreprise en 2008 ; une équipe de recherche dénommée IPSo Alliance s'est dédiée à la promotion du protocole IP (Internet Protocol) pour les réseaux d'objets intelligents miniaturisés.
De nombreuses études ont été menées et ont toutes mis l'accent sur la réalisation optimale de la vision de l'Internet des objets et son aboutissement, malgré tous les défis rencontrés. Cela prend en compte les avancées constantes dans le secteur des appareils intelligents et dans le domaine des technologies de télécommunication, tels que l'informatique en nuage, le concept de réseautage défini par logiciel (SDN), et ainsi de suite [13].

1.2 Définition de l'internet des objets

Le CERP-IOD, "Cluster des projets europeéns de recherche sur l'internet des objets" définit l'internet des objets comme : une infrastructure dynamique d'un réseau global. Ce dernier à des capacités d'auto-configuration basé sur des standards et des protocoles de communication interopérable. Dans ce réseau, les objets physiques et virtuels ont des indentités,des attributs physiques, des personnalités virtuelles et des interfaces intelligentes, et ils sont intégrés au réseau d'une façon transparente [11].

1.3 Domaine d'application de l'internet des objets

L'Internet des objets ne se résume pas seulement à un vaste réseau d'objets intelligents interconnectés et reliés à Internet comme elle montre la figure 1.1, mais représente également et surtout les applications qui constituent véritablement le moteur de cette nouvelle ère de connectivité sur le web.
La présence d'objets intelligents dotés de capacités de communication automatiques et astu-

cieuses va considérablement optimiser le quotidien des individus ainsi que la qualité des services dans divers secteurs grâce à un niveau élevé d'autonomie et d'intelligence.
L'Internet des objets a suscité le développement de nouveaux modèles d'applications sur Internet grâce à ses possibilités.

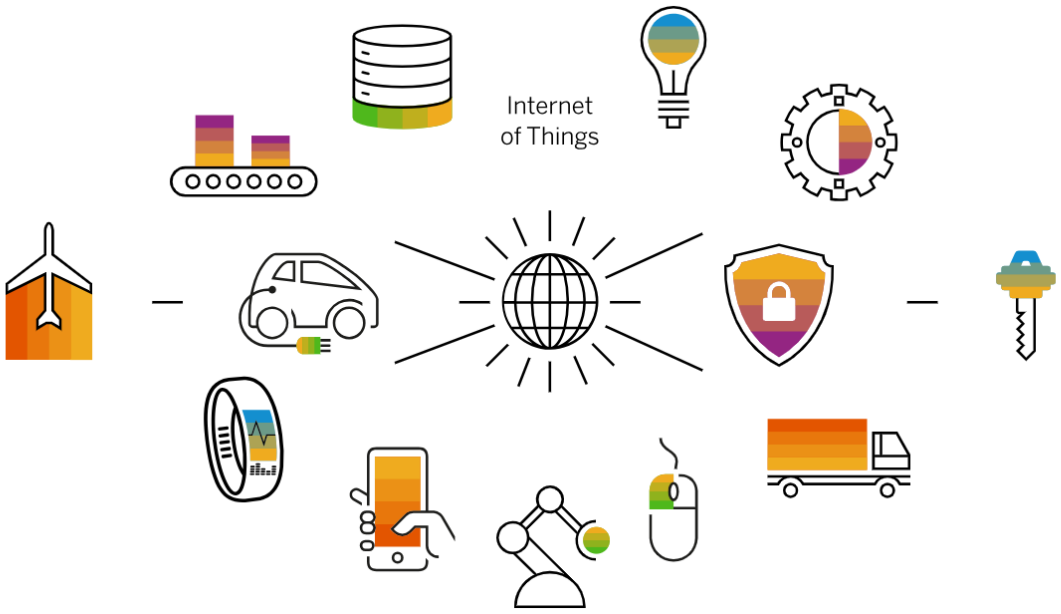

FIGURE 1.1 – Domaine d'application de l'IOT

Dans ce tableau 1.1, nous mettons en avant les applications remarquables de l'IoT.

Domaine	Description
Domotique	La domotique regroupe les méthodes permettant de gérer divers systèmes d'une maison. Elle transforme une maison en un espace intelligent et autonome grâce à l'IoT, qui connecte les appareils domestiques à un réseau pour un contrôle à distance. Son objectif principal est d'améliorer le confort en automatisant ou supervisant des tâches répétitives [5].
Villes intelligentes	Les villes intelligentes désignent un écosystème cybernétique basé sur une infrastructure de communication sophistiquée. Cela optimise l'exploitation des infrastructures urbaines (voies de circulation, réseau électrique, etc.), améliorant ainsi la qualité de vie des résidents [28].
Santé	Des capteurs médicaux permettent de surveiller en direct les paramètres vitaux des patients (température corporelle, tension artérielle, rythme respiratoire). Des capteurs mobiles ou fixes analysent leurs déplacements et leur mode de vie. Ces données sont d'abord traitées localement avant d'être envoyées à des établissements médicaux pour un suivi constant et une réaction rapide si nécessaire [17].
Domaine militaire	Les objets connectés sont déjà employés dans le domaine militaire, notamment avec les drones aériens. Ils se généraliseront jusqu'au niveau du soldat grâce à la miniaturisation et à la baisse des coûts. Ils permettront une connexion entre les échelons de commandement et fourniront des informations précises et instantanées sur l'environnement, la localisation des troupes et leur état physique ainsi que celui de leurs équipements [7].
Sécurité et surveillance	La surveillance sécuritaire est essentielle pour les édifices commerciaux, usines, parkings et espaces publics. Dans le respect de la confidentialité, des capteurs ambiants détectent les substances chimiques nocives. L'analyse des enregistrements vidéo permet d'identifier les comportements suspects [1].

TABLE 1.1 – Applications remarquables de l'IoT

1.4 Composants principaux de l'iot

L'IoT se compose de cinq éléments comme présenté dans le tableau 1.2, avec l'objet connecté étant le principal. Cet objet peut être conçu pour être connecté ou bénéficier d'une connectivité ajoutée ultérieurement.

L'objet recueilli gère les informations provenant de capteurs, les transmet et reçoit des directives pour accomplir une tâche. Pour ces fonctionnalités, une alimentation électrique est généralement requise, surtout si les données sont prétraitées au sein de l'objet [18].

Composant	Description
Capteurs	Dispositifs qui convertissent une grandeur physique (température, pression, etc.) en signal électronique pour analyse et prise de décision.
Actionneurs	Dispositifs qui convertissent une information digitale en action physique pour contrôler des objets ou systèmes (ex. : moteur, vanne, relais).
Énergie	La consommation d'énergie des capteurs est un facteur clé, leur autonomie pouvant aller jusqu'à plusieurs années en fonction de l'optimisation et de la source d'alimentation.
Réseau de capteurs	Un système de capteurs interconnectés, souvent sans fil, permettant la collecte et la transmission de données tout en assurant interopérabilité et accessibilité.
Connectivité	Les objets IoT intègrent une antenne RF pour se connecter aux réseaux et assurer la transmission et la gestion des données.

TABLE 1.2 – Principaux composants de l'IoT

1.5 Architecture générale de l'IoT

Il n'y a pas de consensus universel sur l'architecture de l'IoT qui soit adopté par tous les chercheurs et acteurs dans le monde. Les chercheurs ont suggéré diverses architectures.
D'après certains, l'architecture de l'IoT se divise en trois strates, tandis que d'autres préfèrent une structure à quatre strates. Ils jugent que, compte tenu des progrès de l'IoT, l'architecture en trois couches ne peut satisfaire les besoins des applications. Face aux enjeux de sécurité et de confidentialité dans l'Internet des Objets, une structure en cinq niveaux a aussi été suggérée.

1.5.1 Architecture à trois couches

L'architecture à trois couches de l'IoT présentée dans la figure 1.2 se divise en perception (collecte des données), réseau (transmission) et application (traitement et utilisation) [14].

1. **Couche perception :** La mission primordiale de la couche de perception consiste à identifier des caractéristiques physiques comme la température, l'humidité, le degré d'ensoleillement, la vitesse, et ainsi de suite, via différents capteurs, et à transformer ces données en signaux numériques.
   Les éléments de cette couche peuvent posséder des aptitudes à la détection et/ou des compétences en actionnement [14].
2. **Couche réseau :** Également connue sous le nom de "couche de transmission", elle est chargée d'assurer la liaison et le transfert des données provenant de la couche de perception.Elle sert aussi à la transmission et au traitement des informations provenant des capteurs.
   Les technologies majeures employées pour concevoir cette couche incluent : les technologies cellulaires, WiFi, Bluetooth et Zigbee [14].

3. **Couche application :** Elle est chargée de la gestion des interactions directes avec les utilisateurs finaux.Elle gère les informations provenant de la couche réseau et a pour mission d'offrir à l'utilisateur des services particuliers ainsi que des applications intelligentes [14].

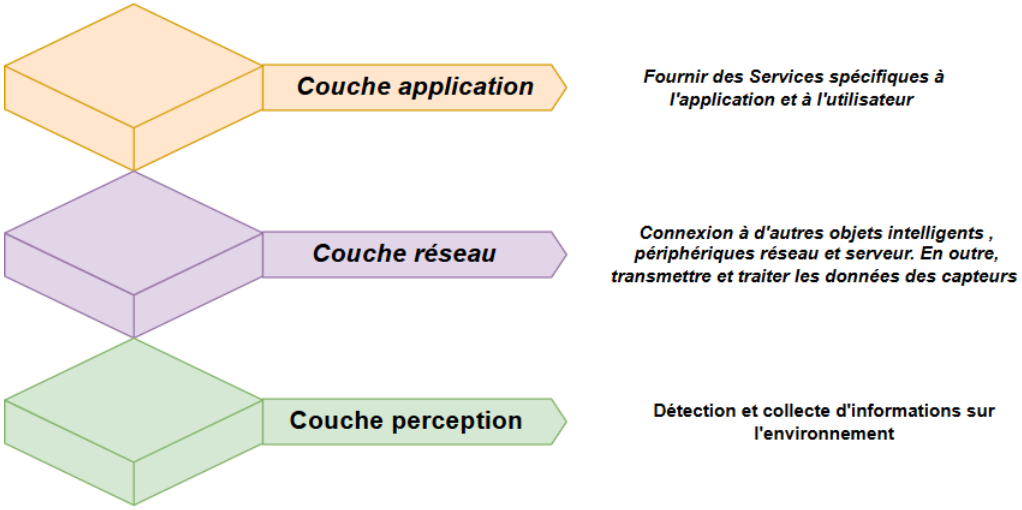

FIGURE 1.2 – L'architecture à trois couches de l'IoT

1.5.2 Architecture à quatre couches

Avec l'évolution constante de l'Internet des Objets, il n'est pas toujours possible de répondre à toutes les exigences. C'est pourquoi une architecture à quatre niveaux a été suggérée. Elle inclut les trois couches de l'architecture précédente, tout en ajoutant une couche supplémentaire nommée couche de support.
L'objectif de la création de cette couche est d'assurer la sécurité.
Dans une structure à quatre strates, les données sont transmises à la couche réseau dérivée de la couche perception [4].

La couche de support : A deux missions à accomplir. Elle atteste que les renseignements sont transmis par un utilisateur authentique et met en œuvre des procédés d'authentification pour contrer les dangers .La seconde responsabilité consiste à transmettre des informations à la couche réseau [4].

1.5.3 Architecture à cinq couches

Le modèle à quatre niveaux a significativement contribué au développement de l'IoT. Le modèle à quatre niveaux a également rencontré des soucis liés à la sécurité et au stockage.
Les chercheurs ont suggéré une structure en cinq couches : la couche de perception, la couche

de transport, la couche d'application, suivies de la couche de traitement et enfin, la couche commerciale [4].

La couche de traitement : souvent désignée comme middleware, elle a pour fonction de rassembler les données issues de la couche de transport. Elle est ensuite chargée de traiter ces informations recueillies et d'éliminer les données inutiles [4].

La couche commerciale : Son rôle principal est de gérer et superviser les applications. En outre, il a la capacité de définir comment créer, stocker et modifier les informations [4].

La figure 1.3 présente les différentes architectures de l'IoT ainsi que leurs couches.

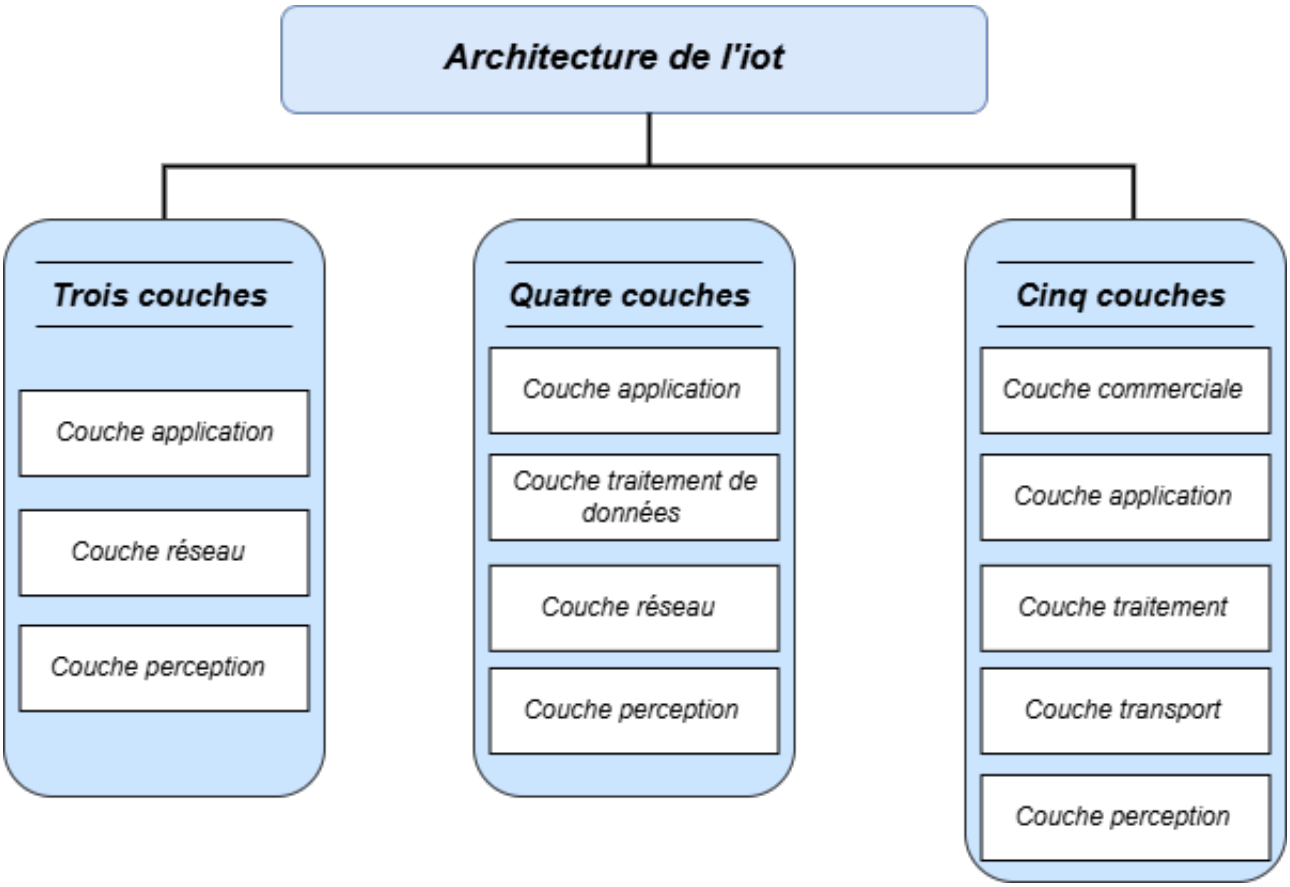

FIGURE 1.3 – L'architecture en couches de l'IoT

L'IoT est un environnement où les objets sont souvent mobiles, les réseaux sont instables et la consommation d'énergie est une contrainte majeure. Le modèle en 5 couches est conçu spécialement pour gérer ces défis.

1.6 Protocoles et technologies de communication

Cette partie analyse les principales technologies et protocoles utilisés en IoT, en mettant en avant leurs caractéristiques, avantages et domaines d'application.

1.6.1 Technologies de communication

L'Internet des objets (IoT) repose sur diverses technologies de communication permettant aux dispositifs de se connecter et d'échanger des données. Voici un aperçu des principales technologies utilisées dans ce domaine [16].

1.6.1.1 Technologies de communication filaires

Les technologies de communication filaires utilisées dans les réseaux IoT sont : Ethernet, HomePlug GP, HomePNA, HomeGrid/G.hn et MoCA [16].

1. **Ethernet (IEEE 802.3) :** Il s'agit d'une technologie de réseau local (LAN) fréquemment employée dans la réalité quotidienne. Ethernet connecte plusieurs appareils à l'aide de câbles en paires torsadées. Cette technologie est plus sûre et moins sujette aux perturbations grâce à son câblage physique. L'Ethernet a été établi comme une technologie filaire susceptible d'être employée dans les résidences intelligentes en offrant des transmissions à la fois sûres et fiables. En outre, il s'agit d'une technologie économique possédant une remarquable capacité de stockage de données [16].

2. **HomePlug Green PHY (GP) :** Il s'agit d'une technologie de transmission de données par voie électrique (PLC). Ce dispositif exploite le filage électrique déjà en place dans un édifice pour relier les équipements de réseau. HomePlug emploie des adaptateurs CPL que l'on insère dans des prises de courant et qui se relient à d'autres dispositifs au moyen de câbles Ethernet. On peut, par exemple, l'utiliser en conjonction avec un réseau Ethernet déjà établi pour relier des dispositifs situés dans une pièce éloignée sans recourir à la technologie sans fil. Le bénéfice majeur du HomePlug GP réside dans son efficacité énergétique [16].

3. **HomePNA (Home Phone Networking Alliance) :** Il s'agit d'une technologie mise en œuvre dans les réseaux domestiques, qui exploite les câbles coaxiaux et téléphoniques existants. Effectivement, les systèmes HomePNA se servent de prises de téléphone, d'adaptateurs HPNA et de ponts combinant des fils coaxiaux et des fils téléphoniques pour établir la connexion entre les dispositifs [16].

4. **HomeGrid/G.hn :** Il s'agit d'une technologie qui exploite les types de câblage domestique existants (câble électrique (PLC), câble coaxial et lignes téléphoniques) ainsi que des fibres optiques en plastique. Un réseau G.hn se sert d'adaptateurs pour relier ses différents équipements. On peut l'utiliser dans les résidences intelligentes, les bureaux et également dans le cadre d'applications industrielles. Il présente des bénéfices supérieurs à HomePNA en raison d'un taux de transfert de données plus élevé [16].

1.6.1.2 Technologies de communication sans fil :

Les technologies de communication sans fil les plus couramment utilisées dans les réseaux IoT sont décrites dans cette section [21].

1. **RFID (Identification par Radiofréquence) :** Il s'agit d'une technologie de communication dont la finalité est de conserver des données en petite taille et, par la suite, de les extraire à distance. Le RFID est fréquemment employé dans les applications de gestion des actifs et de vente au détail, ainsi que pour la traçabilité des stocks. Il peut aussi servir dans un cadre industriel pour repérer aisément et efficacement diverses mesures et informations [21].

2. **NFC (Near Field Communication) :** Il s'agit d'une technologie de communication permettant à deux dispositifs de partager des données via une connexion sans fil à courte portée. Le risque de piratage est diminué grâce à la proximité requise pour que les appareils communiquent via le NFC. C'est pourquoi le NFC est fréquemment employé pour les transactions électroniques [21].

3. **BLE (Bluetooth Low Energy) :** Il s'agit d'une technologie de communication à courte distance employée dans les réseaux personnels (PAN). BLE est une variation de la norme Bluetooth créée pour émettre et recevoir de petites données tout en utilisant une quantité d'énergie minime.En raison de sa facilité d'application, il est couramment employé pour la transmission des données provenant de capteurs. On utilise principalement cette technologie dans les domaines de la domotique, des appareils portables intelligents, de la santé connectée et du commerce de détail [21].

4. **ZigBee (IEEE 802.15.4) :** Il s'agit d'une technologie de portée intermédiaire destinée à des réseaux sans fil économes en énergie. ZigBee est surtout utilisé dans les habitations intelligentes, il propose des opérations solides et extrêmement sécurisées tout en facilitant l'ajout ou la suppression d'éléments du réseau [21].

5. **Z-Wave :** Il s'agit d'une technologie de communication destinée principalement à l'automatisation domestique. Effectivement, il est capable de permettre une communication efficace entre les objets tout en utilisant peu d'énergie. Par ailleurs, Z-Wave simplifie l'intégration d'un nouvel appareil compatible en ajoutant un maillon supplémentaire au réseau [21].

6. **WiFi (Wireless Fidelity) :** Il est couramment utilisé car il permet de relier divers objets entre eux ou directement à Internet. Le WiFi est utilisé de manière efficace pour transférer des fichiers grâce à sa capacité à gérer d'importantes quantités de données.

Elle se distingue principalement par sa vitesse de transfert rapide. Cependant, il présente des limites en matière de portée et de consommation d'énergie. Ses applications les plus optimales se trouvent dans les domaines du Smart Home et du commerce intelligent [21].

7. **LoRaWAN :** Il s'agit d'une technologie LPWAN (réseaux à faible consommation d'énergie et longue portée) qui offre des communications sur de longues distances tout en consommant peu d'énergie. Les technologies LPWAN sont convenablement exploitées dans les applications qui n'exigent pas un débit de données élevé et qui ne sont pas sensibles au temps. On les utilise généralement dans de vastes environnements industriels et commerciaux. Ils sont également utilisés dans les applications de ville intelligente, de construction et d'agriculture [21].

8. **SigFox :** Est une technologie de communication sans fil à faible consommation d'énergie, destinée aux appareils à basse consommation tels que les capteurs et les applications M2M. Elle autorise le transfert de petites quantités d'informations. Cette technologie est couramment employée dans différents domaines comme les compteurs intelligents, les dispositifs de suivi de patients, l'agriculture, les systèmes de sécurité, l'éclairage public et les capteurs environnementaux [21].

9. **NB-IoT (Narrowband IoT) :** Il s'agit d'une technologie LPWAN cellulaire spécialement élaborée pour les réseaux IoT. Il assure une connexion efficace des appareils IoT à travers les réseaux mobiles en gérant de manière sécurisée et fiable de petites quantités de données dans les deux sens. Il est conçu pour échanger de petites quantités de données dans des réseaux comportant un grand nombre d'appareils sur une durée étendue [21].

10. **Technologies mobiles (3G/4G/5G) :** Ces technologies assurent des transmissions de données rapides et une communication sûre sur de vastes étendues. En outre, elles impliquent des coûts d'opération et des besoins en énergie extrêmement élevés [21].

1.6.2 Protocoles de communication

Plusieurs organisations de normalisation ont suggéré diverses normes et protocoles pour l'IoT.

Nous considérons ces derniers comme les plus souvent conseillés et spécifiquement élaborés pour l'IoT.

1. **RPL (Routing Protocol for Low-Power and Lossy Networks) :** Est un protocole de routage conçu spécifiquement pour les réseaux à faible consommation d'énergie et à forte perte, comme ceux utilisés dans l'Internet des Objets (IoT). Il s'appuie sur une approche hiérarchique en construisant un graphe orienté acyclique dirigé (DODAG) pour

organiser la communication entre les nœuds du réseau [9], ce protocole(RPL) sera étudier en détail dans le chapitre qui suit.

2. **MQTT (Message Queue Telemetry Transport) :** Est un protocole de messagerie léger pour les réseaux IoT contraints, optimisé pour la bande passante et l'énergie. Il utilise un courtier central pour router les messages, organisés en sujets, vers les nœuds abonnés du réseau. Il gère les connexions instables avec la fonctionnalité LWT (Last Will and Testament) [21].
3. **AMQP (Advanced Message Queuing Protocol) :** Est un protocole pour la messagerie asynchrone et fiable (publication/souscription) sur TCP, utilisé dans l'IoT pour sa capacité de stockage et de transfert et ses garanties de livraison de messages [27].
4. **CoAP (Constrained Application Protocol) :** Est un protocole léger pour l'IoT contraint qui utilise UDP. Il suit le modèle requête/réponse (REST) et gère la fiabilité des messages. Il permet aux appareils contraints de partager, récupérer et suivre des informations, ainsi que de découvrir d'autres appareils compatibles sur le réseau. Il est adaptable à d'autres types de réseaux[21].
5. **XMPP (Extensible Messaging and Presence Protocol) :**Est un protocole de messagerie et de présence basé sur XML(eXtensible Markup Language), conçu pour l'interopérabilité. Les nœuds sont identifiés par des identifiants uniques et la communication se fait via l'échange de fichiers XML, permettant des structures de données et des requêtes complexes. Bien que flexible et extensible, cette approche XML augmente la complexité et les besoins en ressources (bande passante, calcul) par rapport à d'autres protocoles, ce qui peut être un inconvénient pour les appareils IoT contraints [21].

1.7 Caracteristiques architecturale de l'Internet des Objets

En raison de divers obstacles, développer une application IoT qui réussit reste un défi de taille. Parmi ces défis figurent : la mobilité, la fiabilité, l'évolutivité, la gestion, la disponibilité, l'interopérabilité ainsi que la sécurité et le secret des informations. Nous allons maintenant exposer succinctement chacun de ces défis.

1. **Mobilité :** Les terminaux IoT doivent supporter la mobilité, impliquant une adaptation dynamique de leur adressage IP et de leur configuration réseau en fonction de leur localisation. De plus, cette mobilité peut entraîner des handovers inter-opérateurs, introduisant une complexité accrue liée aux interruptions de service et aux changements de passerelle par défaut [23].
2. **Fiabilité :** Le système doit opérer sans erreur et respecter intégralement ses spécifications. Cette contrainte est fondamentale pour les applications temps réel. Dans l'IoT, la fiabilité et la rapidité de la collecte, de la transmission et du traitement des données sont cruciales ; toute défaillance pouvant avoir des conséquences désastreuses [23].

3. **Scalabilité :** est un défi pour les applications IoT, car de nombreux appareils peuvent être connectés au même réseau, ce qui rend leur gestion complexe. Ces applications doivent donc être conçues pour accepter facilement de nouveaux services et appareils, et permettre des opérations et services extensibles [23].
4. **La gestion :** Piloter un nombre aussi important d'appareils IoT et garder un œil sur leurs problèmes, leurs réglages et leur fonctionnement représente indéniablement une complexité [23].
5. **Disponibilité :** L'IoT met à disposition des utilisateurs des logiciels et du matériel accessibles partout et à tout moment. La disponibilité du logiciel signifie que le service est accessible à toute personne autorisée. La disponibilité du matériel implique que les dispositifs sont facilement utilisables et compatibles avec les caractéristiques et les protocoles IoT. De plus, ces protocoles doivent être compacts pour s'intégrer dans les appareils IoT aux ressources limitées [23].
6. **L'interopérabilité :** L'interopérabilité signifie que les dispositifs et les protocoles hétérogènes doivent pouvoir interagir les uns avec les autres. Ceci est difficile en raison du grand nombre de plates-formes différentes utilisées dans les systèmes IoT [23].

Conclusion

L'Internet des Objets (IoT), qui connecte divers objets physiques à Internet afin d'optimiser le confort de vie et d'améliorer l'efficacité des services, promet des avancées significatives dans de nombreux secteurs. Cependant, cette technologie soulève encore plusieurs défis majeurs, notamment l'interopérabilité, la protection des données et de la vie privée, la consommation énergétique, la disponibilité du service, la mobilité, la diversité des systèmes, l'évolutivité, ainsi que la recherche et la découverte de services.

Pour relever ces défis, il devient indispensable de concevoir des solutions d'optimisation adaptées. Parmi les aspects clés à considérer, le routage occupe une place centrale. C'est pourquoi, dans le chapitre suivant, nous aborderons cette problématique en étudiant les protocoles de routage utilisés dans les réseaux IoT.

2

Protocoles de routage dans l'internet des objets

Sommaire

Introduction

Dans un monde de plus en plus connecté, le routage des données joue un rôle crucial dans le fonctionnement des réseaux informatiques, qu'il s'agisse de l'Internet classique ou des réseaux IoT. Les protocoles de routage permettent d'acheminer efficacement les paquets de données entre les différents nœuds d'un réseau, garantissant ainsi une communication fiable et optimisée.

D'un côté, l'Internet repose sur des protocoles de routage bien établis, comme OSPF et RIP, qui assurent la connectivité et l'échange de routes entre différents réseaux. De l'autre, les réseaux IoT, composés de dispositifs à faible consommation énergétique et souvent soumis à des contraintes de ressources, nécessitent des protocoles adaptés comme RPL.

Ce chapitre explore en détail les protocoles de routage utilisés dans l'Internet et ceux spécifiques aux réseaux IoT. Nous analyserons leurs principes de fonctionnement, leurs différences, ainsi que leurs avantages et limites, afin de mieux comprendre comment ces technologies assurent une communication efficace à l'échelle mondiale et dans les environnements contraints de l'IoT.

2.1 Routage

Le routage est une fonction essentielle des réseaux permettant de sélectionner et d'acheminer les paquets de données d'un nœud source vers un nœud de destination, potentiellement situé sur un réseau distinct. Il repose sur l'analyse des tables de routage pour déterminer le chemin optimal [2].

2.1.1 Étapes et principes fondamentaux d'un routage efficace

Le processus de routage suit une série d'étapes communes, quel que soit le protocole utilisé (comme RIP, OSPF,...) [26]. Ces étapes sont communes à la plupart des protocoles de routage, même s'ils les réalisent de manières différentes :

1. Découverte des voisins : Chaque routeur identifie les nœuds voisins directement connectés.
2. Mesure des coûts des liens : Chaque lien est évalué selon une métrique (latence, bande passante, etc.).
3. Échange d'informations : Les routeurs s'échangent des informations de routage.
4. Calcul des chemins optimaux : Chaque routeur utilise un algorithme (Dijkstra, Bellman-Ford, etc.) pour trouver les meilleurs chemins.
5. Mise à jour de la table de routage : Une table est construite ou mise à jour pour chaque destination (avec next hop et coût).
6. Transmission des paquets : Les paquets sont transmis en suivant la table de routage locale.
7. Convergence : En cas de changement de topologie, les routeurs recalculent les routes pour revenir à un état cohérent.

Pour assurer un acheminement efficace des données à travers un réseau, les protocoles de routage doivent respecter certains principes fondamentaux [26].
Parmi eux on a :

- Optimalité : garantit que les chemins choisis minimisent le coût total.
- La correction : assure que les routes sont valides et sans boucle.
- La stabilité : permet d'éviter les oscillations dues aux changements de topologie.
- La robustesse : rend le protocole capable de s'adapter aux pannes.
- La simplicité facilite sa mise en œuvre.
- Une bonne politique de routage : veille à une utilisation équitable et efficace des ressources du réseau.

Le choix d'une stratégie d'acheminement des paquets dans un réseau repose sur deux grandes catégories : le routage statique, configuré manuellement, et le routage dynamique, ils sont définis comme suit [2] :

- **Le routage statique :** consiste à configurer manuellement les routes sur chaque routeur. Simple et sans surcharge pour les petits réseaux, il devient toutefois difficile à maintenir dans les grandes infrastructures, car chaque modification ou panne nécessite une reconfiguration manuelle, ce qui peut entraîner des erreurs et une lourde gestion.
- **Le routage dynamique :** permet aux routeurs de mettre à jour automatiquement leurs tables de routage. Chaque routeur connaît les réseaux auxquels il est connecté grâce à ses adresses IP et peut vérifier si ses connexions sont actives. Ensuite, il partage ces informations avec les autres routeurs. Peu à peu, chaque routeur est informé de l'état du réseau entier.

Nous nous intéressons au routage dynamique, solution adaptée aux réseaux complexes, par opposition au routage statique réservé aux réseaux simples.

2.2 Protocoles de routage dynamique

Les protocoles de routage dynamiques sont des règles et des méthodes utilisées par les routeurs pour échanger des informations et adapter automatiquement les chemins de communication dans un réseau [2].

Il existe deux types :

- Les protocoles à état de lien.
- Les protocoles à vecteur de distance.

2.2.1 Protocoles à vecteur de distance

Comme son nom l'indique, ces protocoles déterminent le meilleur chemin en évaluant la route la plus courte en termes de distance.
Ils transmettent l'intégralité de leur table de routage à un routeur voisin, c'est-à-dire un routeur directement connecté utilisant le même protocole de routage [2].

Parmi les exemples courants de protocoles à vecteur de distance, on retrouve RIP.

2.2.1.1 Protocole d'information de routage (RIP)

Le Protocole d'information de routage (RIP) est l'un des protocoles de routage intérieur (Protocole de passerelle interne, IGP) les plus anciens et les plus utilisés dans les réseaux informatiques internes. Il permet aux routeurs de s'adapter dynamiquement aux changements de topologie réseau en échangeant des informations sur les réseaux qu'ils peuvent atteindre, ainsi que sur la distance (exprimée en nombre de sauts) qui les en sépare [25].

Développé pour la première fois en 1969 dans le cadre du projet ARPANET, RIP a été largement adopté bien avant l'établissement d'un standard officiel. Il s'agit d'un protocole à vecteur de distance, qui utilise le nombre de sauts comme métrique pour évaluer la distance entre les routeurs [25].

Le nombre maximal de sauts autorisé est limité à 15, ce qui signifie que tout réseau nécessitant plus de 15 sauts est considéré comme inatteignable. RIP fonctionne au-dessus de la couche réseau du modèle Internet et utilise le port UDP 520 pour la transmission de ses messages [25].

Sur le plan algorithmique, RIP repose sur une version distribuée de l'algorithme de **Bellman-Ford**, un algorithme bien connu pour le calcul des plus courts chemins depuis une source unique dans un graphe pondéré, y compris en présence d'arcs de poids négatif. Le caractère distribué de l'algorithme se manifeste par la participation collaborative de plusieurs nœuds au sein d'un système autonome (AS) [25].
Le processus se déroule comme suit :

Algorithme 1 : Algorithme de Bellman-Ford

Entrées : un Graphe valué $G = (V, A, l)$ ($l(a) \in \mathbb{R}$ pour tout $a \in A$)

Sorties : un vecteur π des plus courtes distances issues d'un sommet $s \in V$, et un vecteur p des prédecesseurs des sommets dans ces chemins

```
1  Initialisation : poser π(s) = 0, pour tout v ∈ V \ {s} : π(v) = +∞ et p(v) = Nil ;
2  pour tout i = 1, ..., m − 1 faire
3  |  pour tout (v_i, v_j) ∈ A faire
4  |  |  si π_j − π_i > l(v_i, v_j) alors
5  |  |  |  π_j = π_i + l(v_i, v_j);
6  |  |  |  p(v_j) = v_i
7  |  |  fin
8  |  fin
9  fin
10 pour tout arc (v_i, v_j) ∈ A faire
11 |  si π_j − π_i > l(v_i, v_j);
12 |   alors
13 |  |  il existe un circuit absorbant
14 |  fin
15 fin
16 Renvoyer π, p;
```

- **Explication de l'algorithm**

Cet algorithme 1 cherche les plus courts chemins depuis un sommet source s vers tous les autres sommets dans un graphe valué, même si certaines arêtes ont des poids négatifs. Il commence par initialiser la distance du sommet source à zéro $\pi(s) = 0$, et à l'infini pour tous les autres sommets $\pi(v) = +\infty$. Le vecteur des prédécesseurs p est aussi initialisé avec des valeurs nulles.

Ensuite, l'algorithme effectue $|V| - 1$ itérations (où $|V|$ est le nombre de sommets), et à chaque itération, il parcourt toutes les arêtes $(v_i, v_j) \in A$. S'il trouve qu'un chemin passant par v_i vers v_j est plus court que la distance actuelle $\pi(v_j)$, il met à jour cette distance ainsi que le prédécesseur de v_j.

Enfin, une dernière vérification est effectuée sur toutes les arêtes pour détecter les circuits absorbants (cycles de poids négatif) : si une mise à jour est encore possible à ce stade, cela signifie qu'un tel cycle existe, et donc que les distances minimales ne peuvent pas être correctement définies.

Malgré son ancienneté et sa simplicité, le protocole RIP présente certaines limitations (telles que sa lente convergence et le nombre de sauts limité), ce qui a conduit à son obsolescence progressive au profit de protocoles plus modernes comme OSPF (Open Shortest Path First), basés sur des approches à état de lien (Link-State).

2.2.2 Protocole à état de lien

Le routage à vecteur de distance a été utilisé dans l'ARPANET jusqu'en 1979, lorsqu'il a été remplacé par le routage à état de lien à cause de problèmes de convergence, notamment le problème de "comptage jusqu'à l'infini" [2]. Parmi les protocoles de routage à état de lien, OSPF figure parmi les plus couramment utilisés.

2.2.2.1 Protocole de Routage à Chemin le Plus Court d'Abord (OSPF)

Le routage à Chemin le Plus Court d'Abord (OSPF) est un protocole de routage intérieur (Protocole de passerelle interne, IGP) de type état de lien, largement utilisé dans les réseaux informatiques de grande taille. Il permet aux routeurs de s'adapter efficacement aux changements de topologie en échangeant des informations détaillées sur l'état de leurs liaisons avec les autres routeurs, ce qui leur permet de calculer les chemins les plus courts vers chaque destination du réseau [8].

Introduit pour la première fois à la fin des années 1980 afin de surmonter les limites des protocoles à vecteur de distance comme RIP, OSPF a été normalisé par l'IETF dans le cadre de la suite de protocoles IP. Contrairement à RIP, OSPF n'utilise pas le nombre de sauts comme métrique, mais un coût attribué à chaque liaison, généralement basé sur la bande passante. Ce coût permet une évaluation plus précise de la qualité des chemins empruntés dans le réseau [8].

Chaque routeur OSPF conserve une base de données de l'état de lien contenant une représentation complète et synchronisée de la topologie du réseau. À partir de cette base, le routeur applique **l'algorithme de Dijkstra**, aussi appelé **algorithme du plus court chemin**, pour construire un arbre des plus courts chemins à partir de lui-même vers tous les autres nœuds du réseau [8].

Sur le plan algorithmique, l'algorithme de Dijkstra permet de calculer les plus courts chemins dans un graphe pondéré à partir d'un nœud source donné [8].
Le processus de calcul se déroule comme suit :

Algorithme 2 : Algorithme de Dijkstra

Entrées : un Graphe valué $G = (V, A, l)$ avec $l(a) \geq 0$ pour tout $a \in A$
Sorties : une arborescence T des plus courts chemins et un vecteur π des plus courtances issues d'un sommet $s \in V$

Initialisation : poser $S = \{s\}$, $\pi(s) = 0$, pour tout $v \in V \setminus \{s\}$: $\pi(v) = +\infty$ et $T = \emptyset$;
tant que $S \neq V$ **faire**
  choisir $v_i \in V \setminus S$ tel que $\pi(v_i) = \min\{\pi(v'),\ v' \in V \setminus S\}$;
  $S = S \cup \{v_i\}$;
  pour *tout* $v_j \in \Gamma^+(v_i) \setminus S$ **faire**
    $\pi(v_j) = \min\{\pi(v_j, \pi(v_i) + l(v_i, v_j))\}$;
    si $\pi(v_j) == \pi(v_i) + l(v_i, v_j)$ **alors**
      $T = T \cup \{(v_i, v_j)\}$;
    fin
  fin
fin
Renvoyer π, T;

- **Explication de l'algorithm**

Cet algorithme 2 permet de trouver les plus courts chemins depuis un sommet source s vers tous les autres sommets d'un graphe pondéré à poids positifs ou nuls.

Il commence par initialiser l'ensemble des sommets visités S avec uniquement le sommet source, et attribue une distance $\pi(s) = 0$ à celui-ci, tandis que tous les autres sommets ont une distance initiale infinie $\pi(v) = +\infty$. L'arborescence T des plus courts chemins est initialisée vide.

Ensuite, à chaque itération, l'algorithme sélectionne le sommet v_i non encore visité ayant la plus petite distance estimée. Il l'ajoute à l'ensemble S, puis met à jour les distances de ses voisins v_j non encore visités, en vérifiant si le passage par v_i offre un chemin plus court. Si une meilleure distance est trouvée, elle est mise à jour dans $\pi(v_j)$, et l'arête correspondante est ajoutée à l'arborescence T.

Le processus se répète jusqu'à ce que tous les sommets soient visités, c'est-à-dire que $S = V$. À la fin, l'algorithme renvoie le vecteur des plus courtes distances π ainsi que l'arborescence T représentant les plus courts chemins depuis s.

Grâce à cette méthode, OSPF garantit une convergence rapide, une vision cohérente de la topologie du réseau, ainsi qu'une meilleure efficacité dans le choix des routes.

Les protocoles de routage classiques comme RIP ou OSPF reposent sur des principes bien établis, adaptés à des environnements relativement stables et bien structurés.

Cependant, dans les réseaux IoT, de nouvelles contraintes imposent une refonte ou une adaptation de ces principes.

Ainsi, les protocoles de routage IoT peuvent être vus comme une évolution spécialisée des concepts de routage classiques, adaptée à l'environnement particulier des réseaux d'objets connectés.

Nous allons donc explorer ces protocoles spécifiques, tels que RPL qui répondent aux défis propres à l'IoT.

2.3 Protocoles de routage IOT

Les protocoles de routage IoT assurent une communication efficace et fiable entre les dispositifs connectés, en prenant en compte les contraintes de faible consommation d'énergie et de ressources limitéées. Des protocoles comme RPL et AODV sont spécialement conçus pour répondre aux défis des réseaux de capteurs sans fil et des dispositifs IoT.

2.3.1 Protocole de routage pour les réseaux à faible puissance et avec perte (RPL)

Le RPL est un protocole proactif basé sur un algorithme à vecteur de distance, spécialement conçu pour les réseaux à faible consommation d'énergie et à connectivité instable, appelés réseaux contraints.Ces réseaux sont largement utilisés dans les applications de l'Internet des Objets, telles que les réseaux de capteurs sans fil, les systèmes de surveillance à distance, ou encore les maisons intelligentes. Contrairement aux protocoles classiques, RPL est optimisé pour fonctionner dans des environnements où les ressources sont limitées, que ce soit en énergie, en capacité de traitement, ou en débit, et où les pertes de paquets sont fréquentes [10].

RPL construit une structure arborescente orientée, appelée graphe orienté sans cycle (DODAG), dans laquelle chaque nœud (ou dispositif) définit une ou plusieurs routes vers un point central appelé la racine. Ce point central peut représenter une passerelle vers le réseau global, ou un serveur de collecte de données. Cette structure permet d'organiser efficacement le réseau tout en évitant les boucles de routage, et en s'adaptant aux contraintes du milieu [10].

2.3.1.1 Construction de l'arbre DODAG

Le cœur du fonctionnement de RPL repose sur la création d'un arbre logique de routage nommé DODAG (Graphe orienté acyclique enraciné). La construction de cet arbre débute par la racine, qui initie le processus en diffusant un message d'information contenant des paramètres nécessaires pour rejoindre l'arbre. Ce message est reçu par les nœuds voisins, qui calculent leur

rang en fonction du coût du lien avec la racine. Ensuite, ces nœuds répètent à leur tour ce processus avec leurs propres voisins [10].

Ainsi, chaque nœud choisit un ou plusieurs parents préférés en fonction de critères tels que la qualité du lien, la puissance du signal, ou encore la consommation d'énergie. Ces choix permettent à chaque nœud de déterminer le meilleur chemin vers la racine tout en minimisant les pertes de données et la consommation énergétique. Il est important de noter qu'un nœud peut avoir plusieurs parents, ce qui renforce la fiabilité du réseau en cas de panne ou de changement de topologie [10].

La valeur de rang, qui indique la position d'un nœud dans l'arbre, augmente à mesure qu'on s'éloigne de la racine. Grâce à cette organisation, les données remontent naturellement vers la racine sans risque de boucle. De plus, cette structure permet au réseau de s'auto-organiser : si un lien devient indisponible, le nœud concerné peut rapidement choisir un nouveau parent parmi ses voisins [10].

Dans cette architecture, chaque nœud se voit attribuer une valeur de rang, qui indique sa position relative par rapport à la racine. Plus un nœud est proche de la racine, plus son rang est faible. Cette valeur permet d'organiser le flux des données en direction de la racine, et facilite la détection d'erreurs ou de défaillances dans le réseau [10].

Le protocole RPL prend en charge plusieurs types de communication :

- La collecte de données, où plusieurs nœuds envoient des informations vers un seul point.
- La diffusion d'informations, du point central vers les nœuds.
- La communication entre deux nœuds.

Pour construire et mettre à jour la structure du réseau, RPL utilise trois types de messages :

- Un message (DIO) pour annoncer sa présence et son rang aux autres nœuds.
- Un message (DIS) pour demander des informations aux voisins lorsqu'un nœud arrive dans le réseau.
- Un message (DAO) pour signaler qu'un nœud est joignable, utile notamment pour les communications entre dispositifs.

L'un des grands avantages de RPL est sa capacité à s'adapter automatiquement aux changements dans le réseau. Lorsqu'un lien devient inutilisable ou qu'un nœud est déconnecté, le protocole peut choisir un autre chemin en fonction des autres nœuds disponibles. Cela renforce la résilience et la tolérance aux pannes du réseau [10].

De plus, RPL permet de gérer plusieurs instances de routage dans un même réseau. Chaque instance possède un identifiant unique, ce qui autorise un nœud à participer à plusieurs structures arborescentes indépendantes. Cependant, dans chaque instance, un nœud ne peut appartenir qu'à une seule structure. Cette flexibilité est particulièrement utile dans les systèmes complexes où plusieurs flux de données coexistent [10].

2.3.1.2 Complexité du protocole RPL

Le protocole RPL (Routing Protocol for Low Power and Lossy Networks) est conçu pour minimiser la consommation de ressources dans les réseaux IoT. Sa complexité en termes de trafic réseau dépend fortement du comportement du Trickle Timer, utilisé pour contrôler la fréquence des messages DIO. En régime stable, lorsque le réseau est peu sujet aux changements, le nombre de messages de contrôle est minimal, ce qui confère à RPL une complexité quasi linéaire : $\mathcal{O}(n)$, où n est le nombre de nœuds.

En revanche, dans des environnements dynamiques, la fréquence des réinitialisations du Trickle Timer augmente, ce qui peut faire croître le nombre de messages de manière significative. Dans le pire des cas (forte instabilité topologique), la complexité peut atteindre $\mathcal{O}(n \log n)$ voire plus, en raison de la propagation fréquente de mises à jour dans l'ensemble du DODAG [29].

2.3.2 Ad hoc On-Demand Distance Vector (AODV)

L'AODV est un protocole de routage réactif, c'est-à-dire qu'il établit des routes uniquement à la demande, lorsque des nœuds souhaitent communiquer entre eux. Il est principalement utilisé dans les réseaux ad hoc mobiles, mais il est aussi adopté dans certains scénarios de l'Internet des Objets (IoT), notamment dans les environnements dynamiques où les dispositifs sont mobiles et les ressources limitées [12].

Contrairement aux protocoles proactifs, AODV ne conserve pas constamment des informations de routage pour tous les nœuds du réseau. Cela permet de réduire la consommation de bande passante et de minimiser l'utilisation des ressources, ce qui en fait un choix pertinent pour les réseaux avec faible capacité de traitement ou bande passante limitée, comme c'est souvent le cas dans les réseaux IoT [12].

Le protocole utilise un mécanisme de découverte de route par diffusion. Lorsqu'un nœud souhaite envoyer des données, il diffuse un message de requête (RREQ) dans le réseau pour trouver un chemin jusqu'au destinataire. Une fois la route trouvée, un message de réponse (RREP) est renvoyé au nœud source. Si un lien se casse, AODV peut réparer la route en informant les nœuds concernés grâce à des messages d'erreur (RERR) [12].

Même si AODV est principalement réactif, il contient une petite partie proactive : (les messages hello) échangés périodiquement pour maintenir les routes actives [12].

L'un des avantages majeurs d'AODV est qu'il est sans boucle, grâce à l'utilisation de numéros de séquence qui permettent de gérer efficacement les mises à jour de routes. Cela permet aussi d'éviter le fameux problème du "comptage à l'infini" rencontré dans d'autres protocoles à vecteur de distance [12].

AODV s'adapte rapidement aux changements de topologie dans les réseaux mobiles et IoT, tout en réduisant les besoins en bande passante car il ne nécessite pas de publicités de routage globales. Cependant, il présente aussi des limitations : Une latence plus élevée lors de l'établissement initial de la route et une scalabilité réduite pour les très grands réseaux, à cause de la surcharge liée aux diffusions lors de la découverte de routes [12].

2.3.2.1 Complexité du protocole AODV

AODV est un protocole de routage réactif qui établit les routes uniquement à la demande. La complexité temporelle de la découverte de route varie de $\mathcal{O}(n)$ dans le cas idéal (diffusion rapide), à $\mathcal{O}(n^2)$ dans le pire des cas (réseau dense ou très dynamique). La complexité mémoire est de $\mathcal{O}(r)$, où r est le nombre de routes actives stockées dans la table de routage. En environnement mobile, les ruptures de lien peuvent engendrer des coûts supplémentaires dus aux redécouvertes fréquentes [20].

2.3.3 Comparaison entre AODV et RPL

Dans cette partie nous allons faire une comparaison entre les deux protocoles à savoir AODV et RPL sur plusieurs critères.
Les critères choisis pour comparer ces protocoles de routage dans un réseau IoT ont été sélectionnés car ils couvrent les aspects essentiels liés aux contraintes et exigences de ce type de réseau. Ces éléments sont cruciaux pour garantir un fonctionnement optimal dans des environnements souvent dynamiques, contraints en ressources, et nécessitant une communication stable et sécurisée.

Critère	**AODV**	**RPL**
Type de protocole	Réactif (on-demand)	Proactif (basé sur DODAG)
Famille de protocole	Distance vector	Distance vector (basé sur un graphe orienté sans cycles)
Dynamicité	Dynamique (bien adapté aux environnemets mobiles	N'est pas dynamique
Optimisation énergétique	Faible (nombreux messages de controle)	Élevée (trafic de contrôle optimisé)
Métriques utilisées	Nombres de sauts	Multimétriques
Support des multiples routes	Stocke une seule route valide par destination	Peut maintenir plusieurs routes par destination
Architecture réseau cible	Réseaux Ad-Hoc	Réseaux de capteurs et IOT
Mode de fonctionnement	Route découverte à la demande (RREP, RREP, RERR)	Construction d'un DODAG
Maintenances des routes	Maintenance local avec message RERR	Maintenance proactive avec des messages de controle (DIO, DAO, DIS)
Qualité de service	Non specifiquement conçu pour la QOS	Peut-etre étendu pour supporter la QOS via des métrique(energie, latence)
Complexité de mise en oeuvre	Simple	Plus complexe
Sécurité	Basique, dependante de la couche inferieure	Intégre des options de sécurité (authentification, intéGrité,...)

TABLE 2.1 – Tableau Comparatif entre AODV et RPL

Le tableau 2.1 met en évidence les différences fondamentales entre AODV et RPL, soulignant comment leurs conceptions respectives sont adaptées à leurs environnements cibles. AODV est bien adapté aux réseaux dynamiques et mobiles où la communication est sporadique, tandis que RPL est optimisé pour les réseaux à faible puissance et potentiellement plus statiques comme l'IoT, où l'efficacité énergétique et la gestion proactive des routes sont cruciales.

Conclusion

En conclusion de cette analyse, le routage dans les réseaux sans fil se révèle être un domaine complexe où différents protocoles émergent, chacun avec ses propres forces et faiblesses, adaptés à des scénarios spécifiques. L'étude approfondie des protocoles AODV et RPL illustre parfaitement cette diversité : AODV, avec sa nature réactive, excelle dans la gestion de la mobilité et de la dynamique des réseaux ad hoc, tandis que RPL, par son approche proactive et sa construction de DODAG, répond aux impératifs d'économie d'énergie et de gestion structurée des réseaux de capteurs et de l'IoT. Cependant, les défis persistants liés à la scalabilité, à l'efficacité énergétique dans des environnements mixtes, et à la gestion de la qualité de service nous amènent à considérer si une approche unique peut véritablement satisfaire la complexité des réseaux futurs. C'est dans cette optique que le chapitre suivant explorera la possibilité d'une approche hybride, cherchant à combiner les avantages des mécanismes réactifs et proactifs d'AODV et RPL, dans l'espoir de surpasser leurs limitations individuelles et d'ouvrir la voie à des solutions de routage plus robustes et adaptatives.

3

Algorithme de routage hybride

Sommaire

Introduction

Le routage dans les réseaux de l'Internet des Objets (IoT) constitue un enjeu central, en raison des contraintes spécifiques à ces environnements, telles que la mobilité des nœuds, la limitation des ressources et la variabilité topologique. Ces conditions rendent particulièrement complexe la conception de mécanismes de routage fiables et efficaces. Les protocoles classiques, bien qu'efficaces dans des contextes relativement stables, atteignent rapidement leurs limites face à la dynamique inhérente aux réseaux IoT.

Dans ce chapitre, nous menons une réflexion approfondie autour de la problématique du routage dans les réseaux IoT. Après avoir posé le cadre général de notre étude, nous présentons l'approche proposée pour répondre aux défis identifiés. Celle-ci est ensuite exposée de manière structurée, en détaillant l'algorithme conçu ainsi que les différentes méthodes mises en œuvre pour assurer son fonctionnement.

3.1 Position du problème

Dans l'Internet des Objets (IoT), les nœuds sont en général de petits dispositifs embarqués, conçus pour consommer très peu d'énergie. Ils disposent aussi de ressources limitées en termes de calcul et de mémoire.

Deux grandes contraintes apparaissent alors : D'une part, la consommation d'énergie, qui a un impact direct sur la durée de vie du réseau ; d'autre part, la dynamique de la topologie, causée par la mobilité des nœuds, les défaillances ou les interruptions de connexion.

Ces deux aspects posent un vrai défi aux protocoles de routage, qui doivent garantir des communications fiables entre les objets, tout en restant économes et capables de s'adapter aux changements fréquents dans le réseau.

Or, les protocoles existants montrent leurs limites face à cette double exigence. RPL, par exemple, est efficace sur le plan énergétique, surtout dans des réseaux stables, mais il a du mal à suivre lorsque la topologie évolue rapidement. À l'inverse, AODV est plus flexible face aux changements, mais il consomme beaucoup plus d'énergie à cause de la grande quantité de messages de contrôle qu'il diffuse dans le réseau.

Le vrai défi, donc, est de trouver un bon équilibre : une solution capable de profiter des points forts de RPL et d'AODV, tout en s'adaptant intelligemment à la structure du réseau pour limiter la consommation énergétique et rester réactive face aux changements.

3.2 Solution proposée

Pour répondre aux défis liés à la consommation d'énergie et aux changements fréquents de topologie dans les réseaux IoT, nous avons proposé une solution hybride qui s'appuie sur les points forts de RPL et d'AODV.

Dans cette partie, nous présentons l'algorithme détaillé de l'approche que nous proposons.

Algorithme 3 : l'algorithme de l'approche

Entrées :

— Nœuds avec positions et portée de communication,

— Paramètres de simulation (cycles, coûts énergétiques).

Sorties :

— Mesures de performance (temps, énergie) de l'approche hybride,

— Visualisation du réseau communautaire.

Initialiser le réseau : positionner aléatoirement les nœuds, construire le graphe de connexions, assurer la connexité;
Détecter les communautés : en appliquant l'algorithme de Louvain;
pour *chaque communauté* **faire**
    Simuler la convergence du protocole RPL avec Trickle Timer sur plusieurs cycles;
    Accumuler l'énergie consommée par RPL dans la communauté;
fin
Simuler le protocole AODV : entre représentants des communautés;
Accumuler l'énergie consommée par AODV inter-communautés;
Enregistrer les performances totales;
Afficher la visualisation du réseau :avec colorisation des communautés;
Renvoyer les mesures de performance et la visualisation;

L'algorithme de l'approche hybride proposée combine intelligemment les protocoles RPL et AODV afin d'optimiser le routage dans un réseau IoT.
Il débute par la génération d'un réseau connecté à partir de nœuds positionnés aléatoirement, puis applique l'algorithme de Louvain pour identifier des communautés structurelles.
À l'intérieur de chaque communauté, le protocole RPL est simulé sur plusieurs cycles en tenant compte du Trickle Timer, ce qui permet une diffusion efficace et limitée des messages de contrôle. Ensuite, les communications entre communautés sont assurées par AODV, restreint aux représentants communautaires, réduisant ainsi la portée des échanges inter-nœuds.
L'algorithme cumule l'énergie consommée par chaque protocole et fournit, en sortie, les mesures de performance globales ainsi qu'une visualisation du réseau segmenté en communautés colorées.

3.3 Explication des étapes de l'algorithme

L'objectif général de notre solution et de trouver un équilibre entre l'adabtabilité à la dynamicité du réseau et la consommation de ressources.
Dans ce qui suit nous allons expliquer l'algorithme de l'approche étape par étape :

3.3.1 Première étape

Cette étape consiste à créer un réseau de nœuds connectés, représentant un environnement IoT simulé.

Algorithme 4 : Initialisation du Réseau IoT

Entrées :

— Nombre de nœuds N,

— Zone de déploiement (X, Y),

— Portée de communication R.

Sorties : Un graphe $G = (V, E)$ connexe représentant le réseau IoT.

```
Placer N nœuds aléatoirement dans la zone (X, Y);
pour chaque paire de nœuds (u, v) faire
    si la distance d(u, v) ≤ R alors
        Ajouter une arête (u, v) dans E;
    fin
fin
Construire le graphe G = (V, E) à partir des connexions directes;
si le graphe G n'est pas connexe alors
    Appliquer un algorithme d'arbre couvrant minimal (Kruskal, voir 5) pour connecter
     toutes les composantes;
    Ajouter les arêtes nécessaires dans E pour garantir la connexité;
fin
Renvoyer le graphe G = (V, E);
```

L'algorithme d'initialisation du réseau IoT 4 a pour objectif de générer un graphe connexe représentant un réseau de capteurs déployés aléatoirement dans une zone donnée.
Il commence par placer aléatoirement un nombre déterminé de nœuds dans une zone rectangulaire définie par ses dimensions (X, Y). Ensuite, il examine chaque paire de nœuds et établit une connexion (une arête dans le graphe) si la distance entre ces deux nœuds est inférieure ou égale à une portée de communication prédéfinie R. Une fois ces connexions établies, le graphe est construit.
Cependant, cette construction initiale peut aboutir à un graphe non connexe, c'est-à-dire composé de plusieurs sous-graphes isolés, ce qui est problématique pour le bon fonctionnement des protocoles de routage. Pour pallier cela, l'algorithme vérifie la connexité du graphe et, le cas échéant, applique l'algorithme de Kruskal (voir 5) pour générer un arbre couvrant minimal reliant toutes les composantes déconnectées. Les arêtes issues de cette étape sont alors ajoutées au graphe initial afin d'obtenir une topologie finale pleinement connectée. Ce processus garantit que tous les nœuds du réseau peuvent communiquer, directement ou indirectement, ce qui constitue une condition essentielle pour la mise en œuvre efficace des protocoles de routage dans les réseaux IoT.

Nous allons présenter ici l'algorithme de Kruskal, utilisé pour la construction de l'arbre couvrant minimal dans notre approche.
L'algorithme de Kruskall est une méthode classique et efficace permettant de construire un arbre couvrant de poids minimal à partir d'un graphe connexe et valué. Il repose sur une approche gloutonne qui consiste à trier toutes les arêtes par ordre croissant de poids, puis à les ajouter une à une à l'arbre en construction, à condition qu'elles ne créent pas de cycle. Cette stratégie garantit une solution optimale tout en maintenant une complexité raisonnable [22].

Algorithme 5 : Algorithme de Kruskall

Entrées : un Graphe valué connexe $G = (V, E, p)$ ($p(e) \in \mathbb{R}$ pour toute arête $e \in E$)
Sorties : Un ensemble d'arêtes $T \subset E$ formant un arbre couvrant $H = (V, T)$ de poids minimum de G

```
1  Trier les arêtes par poids croissant. Soient (e_1, e_2, ..., e_n) ordonnées selon cet ordre
   croissant;
2  Initialisation : T = ∅, H = (V, T);
3                   i = 1;
4  tant que card(T) < m − 1 faire
5  |  soit e_i = (u v) ;
6  |  si il n'existe pas de chaîne de u à v dans H alors
7  |  |  T = T ∪ e_i;
8  |  fin
9  |  i = i + 1
10 fin
11 H = (V, T);
12 Renvoyer T;
```

3.3.2 Deuxième étape

Cette étape consiste à regrouper les nœuds du réseau en communautés en appliquant l'algorithme de Louvain, une méthode hiérarchique permettant de maximiser la modularité du graphe[3].
Voici l'algorithme de Louvain utilisé pour détecter les communautés :

Algorithme 6 : Algorithme de Louvain pour la détection de communautés

Entrées : Un graphe pondéré $G = (V, E, w)$

Sorties : Une partition des sommets en communautés maximisant la modularité

1. Initialiser chaque nœud dans sa propre communauté distincte;
2. **tant que** *des améliorations de modularité sont possibles* **faire**
   - **Phase 1 : Optimisation locale de la modularité** ;
   - 3. **pour** *chaque nœud* $i \in V$ *(dans un ordre aléatoire)* **faire**
     - 4. Calculer le gain de modularité ΔQ pour le déplacer dans la communauté de chacun de ses voisins;
     - 5. Déplacer i dans la communauté qui maximise ΔQ, si $\Delta Q > 0$;
   - **fin**
   - 6. Répéter la phase 1 jusqu'à stabilisation (aucun déplacement profitable);
   - **Phase 2 : Construction d'un graphe réduit** ;
   - 7. Construire un nouveau graphe G' où :
     - chaque communauté devient un nœud,
     - les poids des arêtes entre communautés sont la somme des poids entre leurs membres,
     - les liens internes deviennent des boucles.
   - 8. Remplacer G par G' pour l'itération suivante;
3. **fin**
4. 9. Retourner la partition finale des nœuds;

L'algorithme de Louvain 6 pour la détection de communautés est une méthode heuristique efficace qui vise à identifier des groupes de nœuds fortement interconnectés au sein d'un graphe en maximisant une mesure appelée modularité. L'algorithme débute par une phase d'initialisation où chaque nœud du graphe est placé dans sa propre communauté, formant ainsi une partition triviale. Il entre ensuite dans une boucle itérative comportant deux phases principales. La première phase, appelée optimisation locale de la modularité, consiste à examiner chaque nœud pour évaluer si le déplacer dans la communauté de l'un de ses voisins permet d'augmenter la modularité globale. Pour ce faire, le gain de modularité potentiel est calculé pour chaque mouvement possible, et le nœud est déplacé dans la communauté qui offre le meilleur gain positif. Cette phase se répète jusqu'à ce qu'aucun déplacement n'améliore davantage la modularité, marquant une stabilisation locale.

Une fois cette optimisation terminée, la deuxième phase débute : un nouveau graphe réduit est construit. Dans ce graphe, chaque communauté identifiée devient un nœud unique. Les poids des arêtes entre ces nouveaux nœuds correspondent à la somme des poids des liens intercommunautaires dans le graphe original, tandis que les liens internes à une même communauté sont représentés par des boucles. Ce graphe réduit remplace alors le précédent pour une nouvelle itération de l'algorithme, reprenant la phase d'optimisation locale sur cette version agrégée. Le processus s'arrête lorsque plus aucune amélioration significative de la modularité n'est pos-

sible. Finalement, l'algorithme retourne la partition des nœuds en communautés, reflétant une structure modulaire du réseau initial [3].

3.3.3 Troisième étape

Étant un protocole de routage proactif déjà introduit dans le chapitre 2, RPL repose sur la construction d'un DODAG orienté vers la racine [29].
Dans notre approche cet algorithme est appliqué à l'intérieur des communautés.
Nous présentons ci-dessous son algorithme de fonctionnement afin d'en mieux comprendre les mécanismes de convergence et de sélection de routes.

Algorithme 7 : Alorithme du protocole RPL

Entrées : Nœud avec ses voisins, paramètres RPL (Imin, Imax, k, etc.)
Sorties : Parent sélectionné et rang final pour chaque nœud

```
Initialiser rang ← ∞, parent ← Nil, Trickle Timer (I, t, c);
si nœud racine alors
    Rang ← 0, initialiser DODAG, démarrer DIOs;
fin
tant que convergence non atteinte faire
    Décrémenter t;
    si t = 0 et c < k alors
        Envoyer DIO aux voisins, doubler I, réinitialiser t et c;
    fin
    si réception d'un DIO alors
        Calculer rang potentiel;
        si meilleur rang alors
            Mettre à jour rang, parent, réinitialiser Trickle Timer;
            si mode DAO actif alors
                envoyer DAO au parent;
            fin
        fin
        sinon si DIO cohérent du parent actuel alors
            réinitialiser Trickle Timer;
        fin
        sinon
            incrémenter c;
        fin
    fin
fin
```

Cet algorithme 7 décrit le fonctionnement du protocole RPL, conçu pour construire une topologie hiérarchique appelée DODAG au sein d'un réseau IoT. Chaque nœud commence par

initialiser ses paramètres : un rang infini, un parent nul, et un minuteur de Trickle avec les paramètres initiaux (I, t, c).
Le nœud désigné comme racine initie la construction du DODAG en définissant son rang à 0 et en commençant la diffusion des messages DIO. Les autres nœuds attendent la réception d'un DIO avant de procéder à la sélection de leur parent.
Lorsqu'un nœud reçoit un DIO, il calcule un rang potentiel basé sur le rang du nœud émetteur et les coûts associés. Si ce nouveau rang est meilleur que celui déjà enregistré, le nœud met à jour son rang et sélectionne le parent correspondant, tout en réinitialisant le Trickle Timer. Si le mode DAO est actif, le nœud envoie également un message DAO à son parent pour signaler sa position dans l'arbre. Si le DIO reçu provient du parent actuel et est cohérent, le minuteur est simplement réinitialisé pour maintenir la stabilité du DODAG.
En cas de réception de DIOs incohérents ou redondants, un compteur est incrémenté, et le protocole attend que le Trickle Timer atteigne zéro pour potentiellement renvoyer un DIO, sauf si un seuil de redondance (k) est dépassé.
Ce processus se répète jusqu'à ce que tous les nœuds du réseau aient convergé vers une topologie stable, chacun connaissant son rang et son parent dans le DODAG [29].

3.3.4 Quatrième étape

Étant un protocole de routage réactif présenté dans le chapitre 2, AODV établit les routes à la demande uniquement lorsqu'un nœud souhaite communiquer. Il repose sur l'échange de messages de découverte et de réponse pour construire dynamiquement un chemin vers la destination.
Dans notre approche proposée cet algorithme est appliqué entre les communautés.
Nous présentons ci-dessous son algorithme de fonctionnement afin de mieux comprendre ses mécanismes de découverte [20].

Algorithme 8 : Algorithme AODV

Entrées : un Graphe de communication $G = (V, A)$, un nœud source $s \in V$, une destination $d \in V$

Sorties : une route R de s vers d (si elle existe)

Initialisation : chaque nœud initialise sa table de routage à vide ;
chaque nœud note $\pi(v) = +\infty$, sauf s avec $\pi(s) = 0$;
le nœud source s envoie une **DemandeRoute** $(d, 0)$ à tous ses voisins ;
pour *chaque nœud* v_i *recevant une* ***DemandeRoute*** (d, h) **faire**
    si v_i *connaît déjà une route plus optimale vers* d **alors**
        ignorer la demande ;
    fin
    sinon
        enregistrer la route inverse vers l'émetteur ;
        $h \leftarrow h + 1$;
        $\pi(v_i) \leftarrow h$;
        si $v_i = d$ v_i *connaît une route valide vers* d **alors**
            envoyer une **RéponseRoute** (d, h) vers la source via la route inverse ;
        fin
        sinon
            transmettre **DemandeRoute** (d, h) à tous ses voisins sauf l'émetteur ;
        fin
    fin
fin
pour *chaque nœud* v_j *recevant une* ***RéponseRoute*** (d, h) **faire**
    mettre à jour sa table de routage avec la route vers d via l'émetteur ;
    transmettre la réponse en suivant la route inverse jusqu'à s ;
fin
Lorsque s reçoit la première **RéponseRoute**, il établit la route R vers d (plus courte connue) ;
Maintien de Route : pour *chaque nœud* v *détectant une rupture de lien sur une route active* **faire**
    invalider la route correspondante ;
    envoyer un message d'erreur aux nœuds affectés ;
    les nœuds concernés relancent une découverte de route si nécessaire ;
fin
Renvoyer la route R ;

L'algorithme AODV 8 est un protocole de routage réactif conçu pour établir des routes à la demande entre une source et une destination. Le processus commence par l'initialisation de chaque nœud avec une table de routage vide et une distance estimée $\pi(v)$ infinie, sauf pour le

nœud source s dont $\pi(s) = 0$. Le nœud source initie la découverte de route en envoyant un message de demande de route RREQ contenant l'identifiant de la destination et le nombre de sauts h à tous ses voisins. Lorsqu'un nœud v_i reçoit ce message, il vérifie s'il possède déjà une route plus récente ou plus courte vers la destination. Si c'est le cas, il ignore la demande ; sinon, il enregistre l'émetteur comme prochaine étape pour revenir vers la source, met à jour sa propre distance estimée $\pi(v_i) \leftarrow h + 1$, et transmet la demande à ses voisins. Si v_i est la destination ou connaît une route valide vers elle, il génère une réponse de route RREP et la retourne à la source en suivant la route inverse.
Cette réponse est propagée en sens inverse, et à chaque étape, les nœuds mettent à jour leur table de routage avec les informations nécessaires pour atteindre la destination. Une fois que la source reçoit la première réponse, elle considère la route correspondante comme valide et l'utilise pour la communication. En cas de rupture de lien, le nœud concerné invalide la route et envoie un message d'erreur à l'ensemble des nœuds dépendant de cette route. Ceux-ci peuvent alors relancer une nouvelle procédure de découverte si la communication doit se poursuivre [20].

Conclusion

Dans ce chapitre, nous avons abordé la problématique du routage dans les réseaux IoT, en mettant en évidence les principales limites des protocoles traditionnels face aux contraintes dynamiques de ces environnements. Pour y remédier, nous avons proposé une approche hybride fondée sur une détection de communautés : le protocole RPL est utilisé pour gérer efficacement le routage au sein de chaque communauté, tandis qu'AODV assure la connectivité entre les différentes communautés. Cette combinaison vise à tirer parti des atouts respectifs de chaque protocole, afin d'améliorer la robustesse, la scalabilité et l'efficacité globale du réseau.
Dans le chapitre qui suivra, nous nous consacrerons à l'évaluation expérimentale de notre approche. Après avoir présenté le langage de programmation et les bibliothèques utilisées pour sa mise en œuvre, nous procéderons à la simulation, à la discussion détaillée des résultats obtenus, puis à une analyse comparative avec les protocoles RPL et AODV. Cette étape visera à valider l'efficacité de notre solution dans différents contextes de réseau.

4

Évaluations et discussion des résultats

Sommaire

Introduction

Dans ce chapitre, nous décrivons le développement de notre simulation en Python, réalisé sur la plateforme Google Colab à l'aide de bibliothèques essentielles ayant facilité la mise en œuvre de notre approche hybride. Nous nous concentrons sur l'analyse des résultats obtenus dans différents scénarios de simulation, afin d'évaluer les performances de notre solution de routage dans un réseau IoT structuré en communautés. Après avoir présenté les étapes de génération du graphe, de détection des communautés, puis l'application du protocole RPL à l'intérieur de celles-ci et d'AODV pour les communications inter-communautés, nous examinons le temps d'exécution et la consommation énergétique. L'objectif est de comparer notre approche hybride avec les protocoles RPL et AODV classiques pris individuellement, et de mettre en évidence les bénéfices apportés par cette combinaison dans un environnement IoT contraint et évolutif.

4.1 Environnement de développement

Dans cette section, nous avons détaillé les outils, le langage de programmation, la plateforme de développement et les bibliothèques que nous avons exploités pour concevoir et mettre en œuvre notre projet.

4.1.1 Langage de Programmation

Dans ce projet de simulation de protocoles réseau avancés et de détection de communautés, le choix du langage Python s'est imposé naturellement. Sa simplicité de prise en main, combinée à sa puissance, en fait un outil parfaitement adapté pour traiter des problématiques complexes de modélisation et d'analyse de données.
Python est un langage interprété, flexible et polyvalent. Sa syntaxe claire et épurée permet de développer plus rapidement et de traduire des idées complexes en code lisible et maintenable. Il prend en charge plusieurs styles de programmation, comme la programmation procédurale ou orientée objet. Cette dernière étant particulièrement utile pour organiser proprement les différentes composantes de la simulation.
Python gère aussi automatiquement la mémoire et utilise un typage dynamique, ce qui facilite grandement le prototypage et les ajustements en cours de développement. Son principal atout reste sans doute la richesse de son écosystème : entre les bibliothèques standards et les nombreux modules tiers, on dispose d'outils puissants pour manipuler des graphes, faire des calculs scientifiques ou visualiser les résultats. Tous ces éléments font de Python un environnement de choix pour mener des recherches et concevoir des systèmes complexes dans le domaine des réseaux [15].

FIGURE 4.1 – Logo python

4.1.2 Plateforme de Développement

Google Colaboratory, ou Colab, a été un outil central pour la conceptualisation et l'implémentation de notre projet. Cette application web gratuite, basée sur le cloud et compatible avec Python, offre un environnement de type Jupyter Notebooks puissant, permettant de combiner code exécutable, visualisations interactives et documentation explicative au sein d'un même espace de travail collaboratif. Ses capacités intégrées d'exécution de code Python, sa prise en

charge des sorties riches en texte et graphique, ainsi que l'accès gratuit à des ressources matérielles accélérées telles que les GPU et les TPU, en font un choix particulièrement judicieux. Colab se distingue également par sa simplicité, ne nécessitant aucune configuration locale et intégrant des bibliothèques Python préinstallées, ce qui facilite grandement le prototypage rapide et la mise en œuvre efficace de modèles, notamment dans les domaines de la science des données dans divers domaines. L'utilisation de Google Colab a ainsi significativement fluidifié l'implémentation, le débogage et la visualisation intuitive des résultats de notre modèle [19].

FIGURE 4.2 – Logo Google Colab

4.1.3 Bibliothèques utilisées

Les bibliothèques suivantes ont été utilisées dans notre projet :

4.1.3.1 Manipulation de données et calculs numériques

- **NumPy :** Une bibliothèque Python essentielle pour le calcul numérique de haute performance, offrant des objets de tableaux multidimensionnels et une vaste collection de routines pour traiter ces tableaux.

4.1.3.2 Visualisation de données

- **Matplotlib :** Une bibliothèque de traçage complète pour créer des visualisations statiques, animées et interactives en Python.

4.1.3.3 Analyse et Modélisation de Graphes

- **NetworkX :** Une bibliothèque Python fondamentale pour la création, la manipulation et l'étude de la structure, de la dynamique et des fonctions de réseaux complexes.

4.1.3.4 Utilitaires de simulation

- **Random :** La bibliothèque standard de Python pour générer des nombres pseudo-aléatoires.

- **Time :** Une bibliothèque standard de Python qui fournit des fonctions liées au temps. Elle est utilisée spécifiquement pour mesurer la durée d'exécution des différentes phases de la simulation.

4.2 Résultats de l'approche hybride

Dans cette section, nous présentons et analysons en détail les résultats obtenus suite à la simulation de notre approche, visant à concevoir un réseau IoT adapté à la mobilité des nœuds tout en réduisant la consommation d'énergie.
Pour mener à bien notre évaluation et obtenir les résultats, nous avons développé un programme en Python permettant d'implémenter l'algorithme présenté dans le chapitre 3. Ce programme a été utilisé pour simuler des réseaux IoT de tailles variées, allant de petites configurations à des réseaux à grande échelle.

4.2.1 Première phase

Dans cette phase, comme illustré dans l'Algorithme 4, un graphe initial est généré, composé d'un nombre nœuds répartis aléatoirement dans une grille de taille limitée.
Les arêtes du graphe sont établies entre chaque paire de nœuds dont la distance est inférieure ou égale à une portée de communication fixée, ce qui permet de simuler la connectivité locale. Pour garantir que le graphe soit entièrement connecté un arbre couvrant minimal est ajouté à l'aide de l'algorithme de Kruskal (voir 5).
Pour visualiser ce graphe, nous avons utilisé la bibliothèque NetworkX pour la modélisation et Matplotlib pour l'affichage.

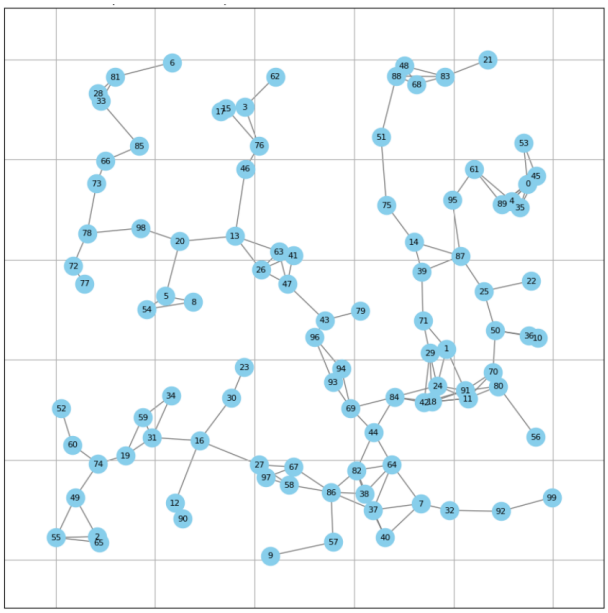

FIGURE 4.3 – Visualisation d'un réseau IoT de 100 nœuds

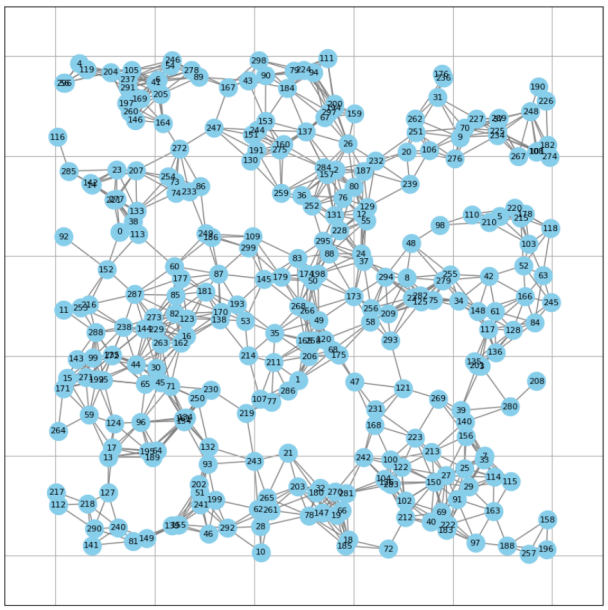

FIGURE 4.4 – Visualisation d'un réseau IoT de 300 nœuds

4.2.2 Deuxième phase

Après avoir généré un graphe dans la phase 1, nous passons à la phase 2, où ce graphe est partitionné en communautés à l'aide de l'algorithme de Louvain, présenté dans l'algorithme 6. Cette étape nous permet d'obtenir une structure communautaire, représentée sous forme de sous-graphes distincts. Le résultat est visualisé sous forme d'un graphe coloré, chaque couleur correspondant à une communauté différente, comme illustré dans les figures suivantes.

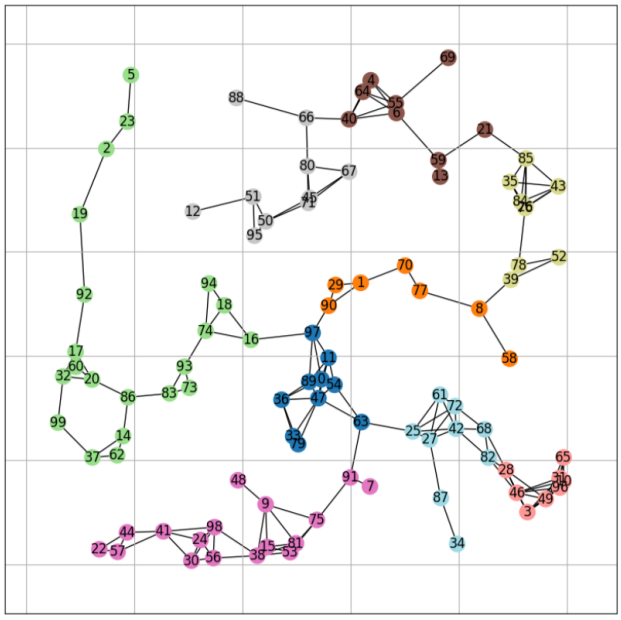

FIGURE 4.5 – Visualisation d'un réseau IoT de 100 nœuds partitionné en communautés

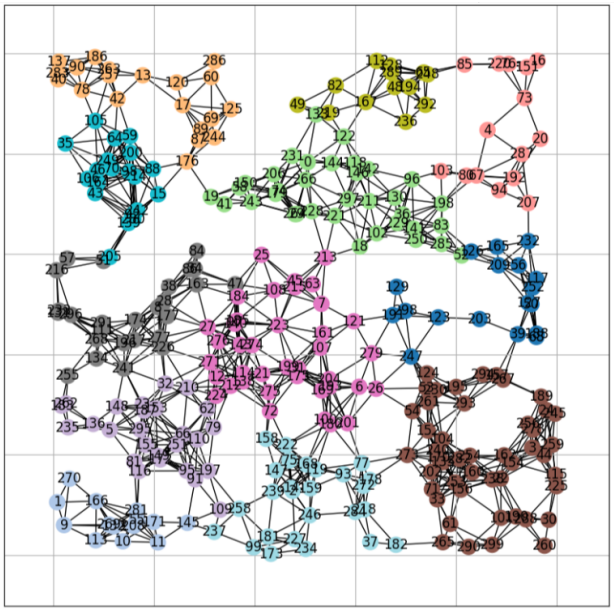

FIGURE 4.6 – Visualisation d'un réseau IoT de 300 nœuds partitionné en communautés

4.2.3 Troisième phase

À cette phase, après avoir divisé le graphe en communautés à l'aide de l'algorithme de Louvain (décrit dans l'algorithme 6), nous appliquons le protocole RPL (illustré dans l'algorithme 7) à l'intérieur de chaque communauté. Pour le routage entre les communautés, nous utilisons le protocole AODV(illustré dans l'algorithme 8). Cette approche vise à obtenir un réseau plus adapté aux changements de topologie et à la mobilité des nœuds, tout en minimisant la consommation d'énergie. Nous mesurons à la fois l'énergie consommée et le temps d'exécution à l'aide de la bibliothèque time de python.
Après avoir simulé le fonctionnement de ce réseau hybride, nous avons obtenu les résultats suivants :

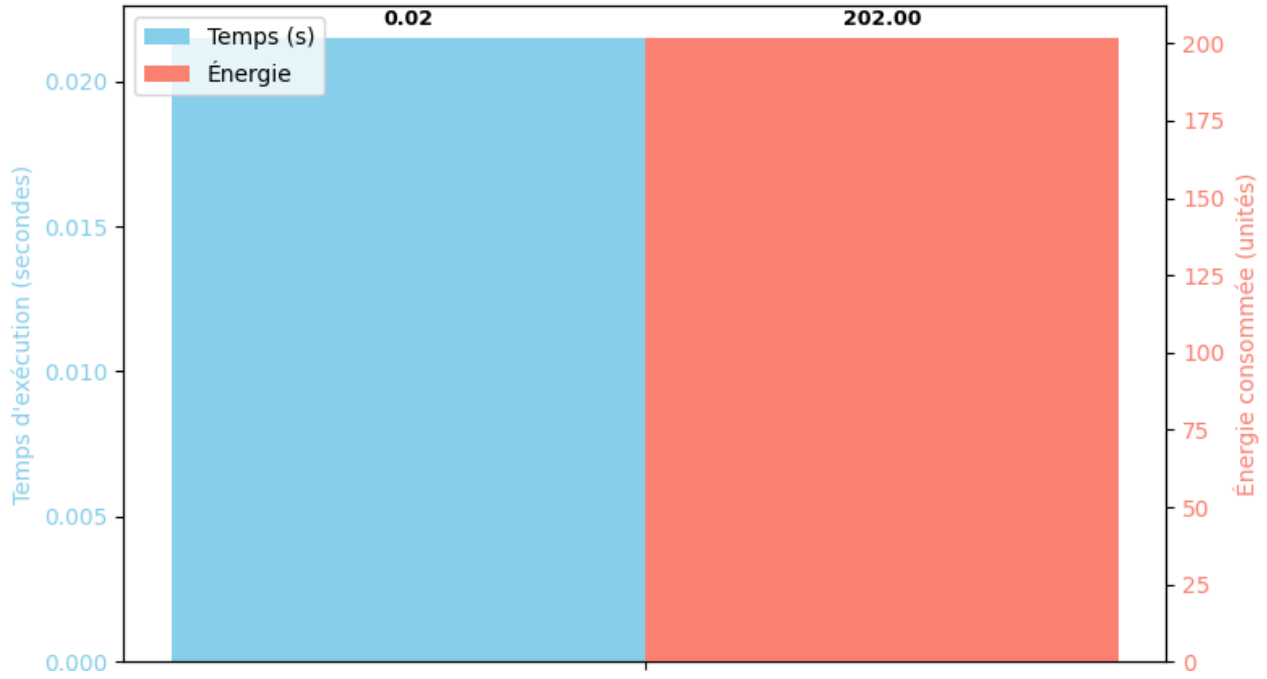

FIGURE 4.7 – Performances du routage hybride sur un réseau IoT de 100 nœuds

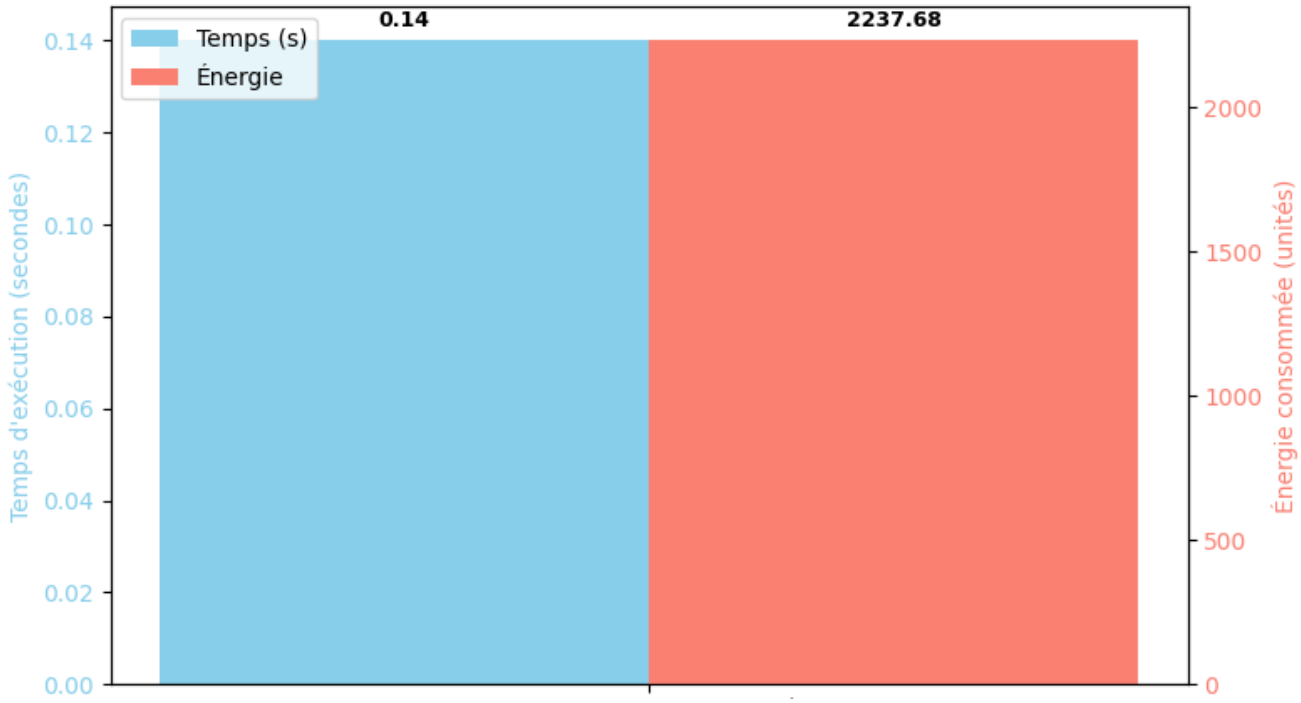

FIGURE 4.8 – Performances du routage hybride sur un réseau IoT de 300 nœuds

4.2.4 Discussion des résultats de l'approche hybride

Les simulations menées sur des réseaux de tailles différentes révèlent des tendances claires concernant la consommation d'énergie et le temps d'exécution de l'approche hybride.

4.2.4.1 Performance pour 100 Nœuds

- Temps d'exécution : 0.02 secondes ;
- Énergie consommée : 202 unités.

Pour un réseau de petite à moyenne taille, l'approche hybride présente une performance remarquable en termes de temps d'exécution, atteignant seulement 0,02 seconde. Cette efficacité témoigne de la rapidité de la phase de détection des communautés, de la construction des routes RPL au sein des communautés, ainsi que de la gestion optimisée des routes inter-communautés via AODV. La consommation énergétique, estimée à 202 unités, correspond au coût initial associé à l'établissement de la topologie et à l'échange des messages de contrôle nécessaires à la stabilisation du routage.

4.2.4.2 Performance pour 300 Nœuds

- Temps d'exécution : 0.14 secondes.
- Énergie consommée : 2237.68 unités.

Pour un réseau de grande taille, l'approche hybride continue de démontrer une scalabilité satisfaisante malgré l'augmentation significative des ressources sollicitées. Le temps d'exécution atteint 0,14 seconde, contre 0,02 seconde pour 100 nœuds, soit une augmentation par un facteur de 7. Néanmoins, ce temps reste très raisonnable compte tenu de la complexité accrue du réseau, ce qui confirme l'efficacité temporelle de l'approche même à grande échelle. L'algorithme de détection de communautés (Louvain) ainsi que les mécanismes de routage intra-communautaire (RPL) et inter-communautaire (AODV) parviennent à gérer efficacement cette montée en charge.

En ce qui concerne la consommation d'énergie, une hausse notable est observée, passant de 202 à 2237,68 unités, soit une augmentation d'environ un facteur 11. Cette élévation s'explique par la multiplication des messages de contrôle (DIO, DAO) échangés au sein des communautés RPL et des messages AODV transmis entre les représentants inter-communautaires. Chaque opération de transmission et de réception contribue directement au coût énergétique global, ce qui est attendu dans un réseau plus dense et plus complexe.

4.3 Analyse Comparative

Afin d'évaluer les performances de l'approche hybride proposée, nous l'avons comparée aux deux protocoles classiques utilisés dans les réseaux IoT : RPL en mode global (où l'ensemble du réseau forme un unique DODAG) et AODV en mode global (où les routes sont établies à la demande entre tous les nœuds). Cette comparaison repose sur deux critères essentiels : le temps d'exécution, qui reflète la rapidité de la configuration et du routage, et la consommation d'énergie, qui est un facteur critique dans les réseaux IoT contraints.

Les tests ont été effectués sur des topologies de 100 et 300 nœuds afin d'analyser le comportement de chaque approche à différentes échelles. Les résultats obtenus permettent de mettre en évidence les avantages et les limites de chaque protocole dans un contexte IoT dynamique.

Les figures suivantes (4.9 et 4.10) illustrent les résultats obtenus lors des simulations, en comparant les performances des trois (l'approche hybride, RPL global et AODV global) en termes de temps d'exécution et de consommation d'énergie pour des réseaux de 100 et 300 nœuds.

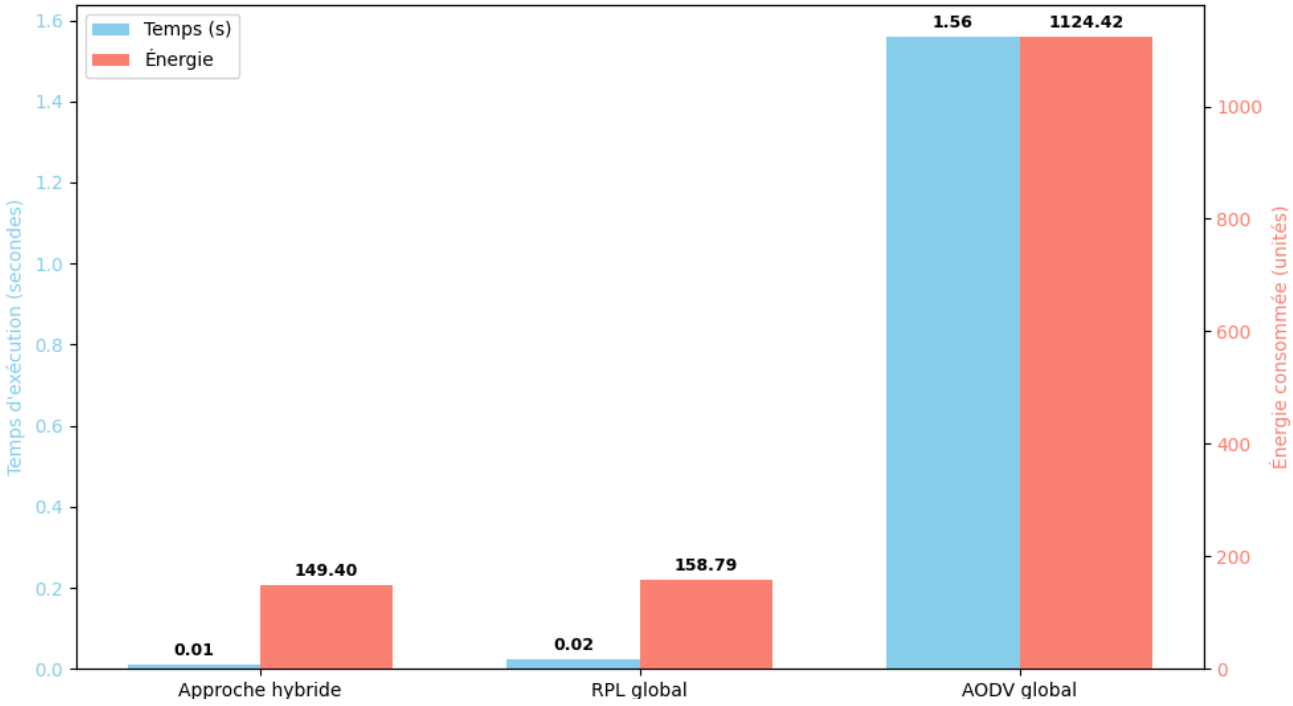

FIGURE 4.9 – Comparaison des performances de l'approches Hybride, RPL global et AODV global sur un réseau IoT de 100 nœuds

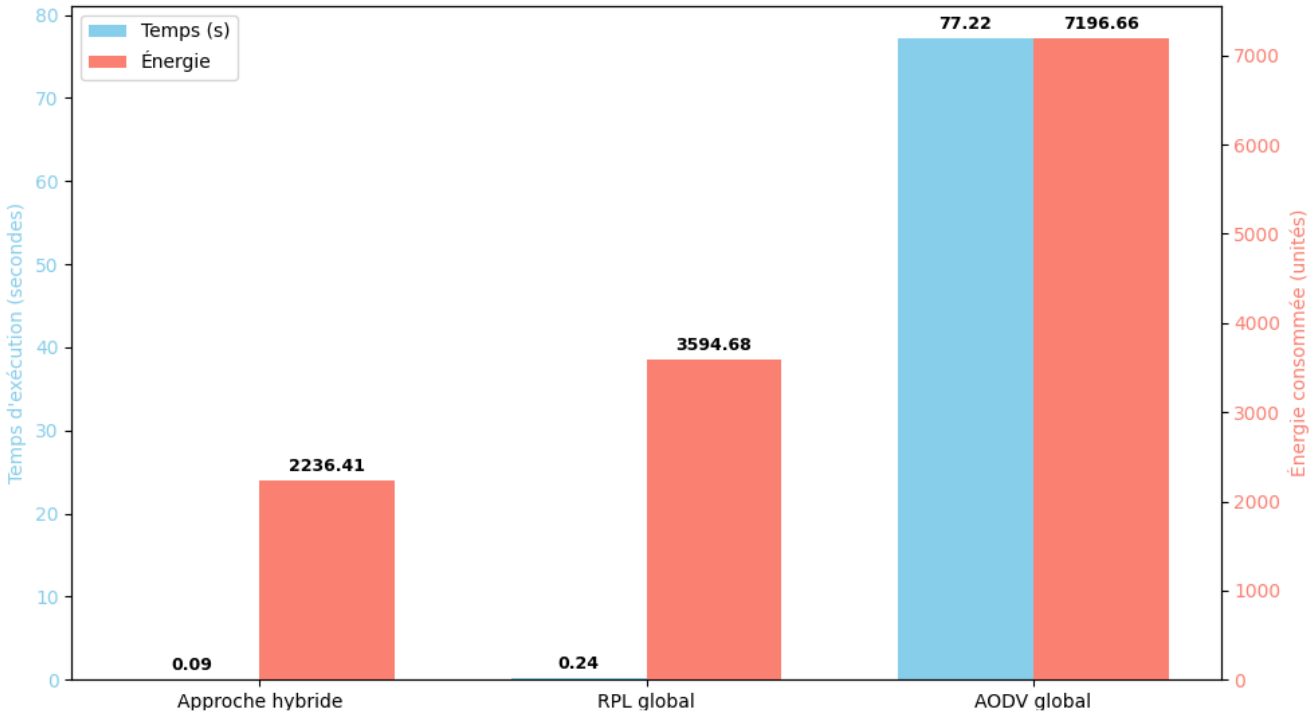

FIGURE 4.10 – Comparaison des performances de L'approches Hybride, RPL global et AODV global sur un réseau IoT de 300 nœuds

4.3.1 Interprétation des résultats de simulation

Les figures(4.9 et 4.10) présentent une comparaison des performances des trois approches : l'approche hybride, le RPL global et l'AODV global, appliquées à des réseaux IoT de 100 et 300 nœuds. Deux critères d'évaluation sont analysés : le temps d'exécution et la consommation énergétique.

4.3.1.1 Performance d'un réseau IoT de 100 Nœuds

Dans un réseau de petite taille comportant 100 nœuds, les résultats de simulation montrent que l'approche hybride est la plus performante. Elle affiche un temps d'exécution très réduit de 0,01 seconde, contre 0,02 seconde pour le RPL global et 1,56 seconde pour l'AODV global. Du point de vue énergétique, elle consomme 149,40 unités, ce qui est légèrement inférieur au RPL (158,79 unités) et nettement inférieur à l'AODV (1124,42 unités).
Ces performances s'expliquent par l'organisation en communautés, qui limite la portée des messages de routage RPL à l'intérieur des sous-graphes, tandis que les messages AODV inter-communautés sont restreints à un sous-ensemble du réseau. À cette échelle, le RPL global reste relativement compétitif, mais l'AODV global montre ses limites, notamment par une consommation énergétique excessive et un temps de réponse élevé, dus à l'inondation des requêtes dans tout le réseau.

4.3.1.2 Performance d'un réseau IoT de 300 Nœuds

Lorsque la taille du réseau est portée à 300 nœuds, les écarts de performance deviennent beaucoup plus marqués. L'approche hybride maintient des performances optimales avec un temps d'exécution de 0,09 seconde et une consommation énergétique de 2236,41 unités. Le RPL global, quant à lui, reste fonctionnel avec un temps d'exécution de 0,24 seconde, mais subit une hausse significative de la consommation énergétique (3594,68 unités), en raison de l'augmentation du nombre de messages de contrôle échangés dans le DODAG.
L'AODV global en revanche, voit ses performances se dégrader drastiquement : le temps d'exécution atteint 77,22 secondes et la consommation énergétique dépasse 7196,66 unités. Ces résultats illustrent clairement le manque de scalabilité de l'approche AODV en environnement dense, dû à son mécanisme de diffusion de requêtes dans l'ensemble du réseau.

Afin de mieux évaluer l'efficacité énergétique des différentes stratégies de routage étudiées, une analyse comparative basée sur la complexité énergétique a été réalisée. Trois approches ont été considérées : l'approche hybride combinant RPL et AODV au sein de communautés, le RPL global, et l'AODV global.

Stratégie	Complexité énergétique
RPL + AODV (communautés)	O(C * k² + C²)
RPL global	O(N²)
AODV global	O(N²)

FIGURE 4.11 – résultat de complexité

La figure 4.11 présente une comparaison des complexités énergétiques des trois stratégies de routage utilisées dans cette étude :

Les protocoles RPL global et AODV global présentent tous deux une complexité énergétique de l'ordre de $\mathcal{O}(N^2)$, où N désigne le nombre total de nœuds dans le réseau. Dans le cas de RPL, cette complexité s'explique par la gestion d'un DODAG unique, dans lequel chaque modification de la topologie peut entraîner une large diffusion de messages de contrôle à travers l'ensemble du réseau. Quant à AODV, sa complexité élevée résulte de son mécanisme réactif de découverte de routes, qui repose sur la diffusion de requêtes auprès de tous les nœuds à chaque nouvelle demande de communication.

À l'inverse, l'approche hybride, qui divise le réseau en C communautés regroupant chacune environ k nœuds, présente une complexité énergétique nettement plus avantageuse, estimée à $\mathcal{O}(C \cdot k^2 + C^2)$.
Cette structuration permet de réduire la portée des échanges et en limitant la redondance des messages de contrôle. Elle se révèle donc plus efficiente, mais aussi plus optimale en termes de

consommation d'énergie, dans les réseaux de grande taille, où la scalabilité constitue un enjeu majeur.
Les résultats obtenus confirment la supériorité de cette approche par rapport aux solutions classiques, aussi bien sur le plan théorique que pratique.

Conclusion

En conclusion, les simulations menées ont permis de valider l'efficacité de notre approche hybride en la comparant aux protocoles classiques RPL et AODV. Les résultats obtenus montrent clairement que notre solution surpasse ces derniers en termes de consommation énergétique et de temps d'exécution. Elle répond de manière pertinente aux limitations précédemment identifiées dans les protocoles traditionnels.
Ces performances confirment la pertinence de notre démarche dans des environnements IoT dynamiques et contraints, et ouvrent la voie à des perspectives d'optimisation plus poussées pour des réseaux à plus grande échelle ou plus hétérogènes.

Conclusion générale

Dans ce mémoire, nous nous sommes intéressés à la problématique du routage dans les réseaux de l'Internet des Objets (IoT), un domaine marqué par des contraintes fortes telles que la mobilité des nœuds, la limitation des ressources énergétiques, et la dynamique topologique constante. Face aux limites des protocoles classiques, nous avons proposé une approche hybride combinant la robustesse du protocole RPL pour le routage intra-communautés, et la réactivité d'AODV pour les communications inter-communautaires.

Notre démarche a reposé sur l'intégration de l'algorithme de Louvain pour la détection de communautés, permettant une structuration logique du réseau et une réduction significative de la portée des messages de routage. La simulation de cette approche dans un environnement Python (Google Colab) a permis de mesurer ses performances, notamment en termes de consommation d'énergie et de temps d'exécution, et de les comparer aux protocoles RPL et AODV utilisés isolément.

Les résultats obtenus ont mis en évidence l'efficacité de notre solution hybride, qui parvient à répondre aux principales limitations identifiées dans les approches classiques. L'adaptation dynamique, la limitation de la propagation inutile des messages, et la gestion optimisée des ressources démontrent la pertinence de l'approche proposée pour des scénarios IoT réels et évolutifs.

Ce travail a permis d'approfondir notre compréhension des mécanismes de routage dans les réseaux de l'Internet des Objets (IoT), en particulier face aux contraintes dynamiques et à la structure communautaire des nœuds. L'approche hybride proposée, combinant RPL et AODV, s'est révélée efficace en termes de consommation énergétique et de temps d'exécution. Les résultats obtenus ouvrent la voie à de nouvelles pistes de recherche, que ce soit dans l'optimisation des protocoles de routage, l'intégration de l'intelligence artificielle pour des décisions adaptatives, ou encore l'application de cette approche à des contextes concrets tels que les villes intelligentes, les réseaux de capteurs environnementaux ou les systèmes critiques nécessitant une communication fiable et économe.

Bibliographie

[1] Yanis Atoumi and Sonia Bensadi. Approche évolutionnaire pour la composition de services sensible à la qos dans l'internet des objets à large échelle. Master's thesis, Université A. Mira de Béjaia, Béjaia, 2018.

[2] Zaheer Aziz, Johnson Liu, Abe Martey, and Faraz Shamim. *Troubleshooting IP Routing Protocols*. 2002.

[3] Vincent Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics : Theory and Experiment*, 2008(10) :P10008, 2008.

[4] Muhammad Burhan, Bilal Khan, and Byung-Seo Kim. IoT elements, layered architectures and security issues : A comprehensive survey. *Revue Internet des Objets*, 37, 2018.

[5] Chapman, Hall/CRC, Taylor, and Francis. *Internet of Things : Challenges, Advances, and Applications*. 2018.

[6] Simone Cirani, Gianluigi Ferrari, Marco Picone, and Luca Veltri. *Internet of Things : Architectures, Protocols and Standards*, volume 403. 2019.

[7] Lamine Djebri and Mourad Lamraoui. Prévision de stationnement dans un environnement IoT. Master's thesis, Université Akli Mohand Oulhadj - Bouira, Bouira, 2019.

[8] Jeff Doyle and Jennifer DeHaven Carroll. *Routing TCP/IP, Volume 1*. 2005.

[9] Olivier Hersent, David Boswarthick, and Omar Elloumi. *The Internet of Things : Applications to the Smart Grid and Building Automation*. 2012.

[10] Ali Kadhum Idress and Athraa J. H. Witwit. A comprehensive review for (RPL) routing protocol in low power and lossy networks. *Conference Paper in Communication and Information Science*, 2018.

[11] Saleh Imad. Les enjeux et les défis de l'internet des objets (IOD). *Revue Internet des Objets*, 19, 2017.

[12] Shubha Jain, Rahul Singh, and Praveen Kumar Tripathi. Discussion of ad-hoc on-demand distance vector (AODV) routing protocol for mobile host. *International Journal of Computer Applications*, 2010.

[13] Nadjah Kara. Conception d'un réseau de communication pour une maison intelligente en utilisant la technique d'internet des objets. Master's thesis, Université A. Mira - Béjaia, Béjaia, 2017.

[14] Tinhinane Khaldi and Cylia Khelifi. Étude d'impacts des attaques sur le protocole de routage RPL dans l'IoT. Master's thesis, Université A. Mira de Béjaïa, Béjaia, 2023.

[15] Mark Lutz. *Learning Python, Fifth Edition*. 2013.

[16] Sami Mnasri and Val Thierry. The internet of things enabling communication technologies, applications and challenges. *International Journal of Wireless and Mobile Computing*, 13, 2022.

[17] Younes Ait Mouhoub and Fatah Bouchebbah. Proposition d'un modèle de confiance pour l'internet des objets. Master's thesis, Université A. Mira de Béjaïa, Béjaia, 2015.

[18] Belhadj Naceur and Abbad Abdelhak. La sécurité de l'internet des objets (IoT). Master's thesis, Université Ibn-Khaldoun de Tiaret, Tiaret, 2022.

[19] Poornima G. Naik, Girish R. Naik, and M. B. Patil. *Conceptualizing Python in Google COLAB*, volume 330. 2022.

[20] Charles Edward Perkins, Elizabeth Michele Belding-Royer, and Samarjeet Singh Das. Ad hoc on-demand distance vector (AODV) routing. *RFC 3561*, 2003.

[21] Octavian Adrian Postolache, Edward Sazonov, and Subhas Chandra Mukhopadhyay. *Sensors in the Age of the Internet of Things : Technologies and applications*. The Institution of Engineering and Technology, 2019.

[22] Michel Sakarovitch. *Optimisation combinatoire : méthodes mathématiques et algorithmiques. Volume 1 : Graphes et programmation linéaire*. Hermann, Collection "Enseignement des sciences", 1984.

[23] Tara Salman and Raj Jain. Networking protocols and standards for internet of things. *Computer Networks and Internet Research*, 27, 2017.

[24] Peter Sennhauser. *L'internet des objets*. SUISSEDIGITAL et l'École supérieure d'économie de Zurich HWZ, 2018.

[25] Andrew Stuart Tanenbaum, Nicholas Feamster, and David James Wetherall. *Computer Networks*. 2012.

[26] Andrew Stuart Tanenbaum, Nicholas Feamster, and David James Wetherall. *Computer Networks*. 2021.

[27] Tanya Mohan Tukade and Rajeshwari Banakar. Data transfer protocols in IoT : An overview. *International Journal of Pure and Applied Mathematics*, 7, 2018.

[28] John Wiley and Sons. *Internet of Things*. 2018.

[29] Timothy Winter, Pascal Thubert, Anantho Brandt, Jonathan Hui, Robert Kelsey, Philip Levis, Kris Pister, Rolf Struik, and Jean-Pierre Vasseur. RPL : IPv6 routing protocol for low-power and lossy networks. *RFC 6550*, 2012.

Résumé

Cette thèse porte sur l'optimisation du routage dans les réseaux de l'Internet des Objets (IoT), en s'intéressant particulièrement à l'intégration d'approches communautaires et de stratégies de routage hybrides. Les réseaux IoT, composés de nœuds souvent contraints en énergie et en ressources, présentent de nombreux défis tels que la mobilité, la scalabilité, la fiabilité des communications et la gestion des contraintes dynamiques.

L'étude s'articule autour de l'idée que l'organisation communautaire des nœuds peut renforcer l'efficacité du routage. Ainsi, une approche originale est proposée, combinant l'algorithme de détection de communautés de Louvain avec des protocoles de routage classiques comme RPL (adapté à l'IoT) pour la communication intra-communautaire, et AODV (plus dynamique) pour la communication inter-communautaire. Chaque étape de cette méthode a été détaillée, implémentée et testée.

Les expérimentations, réalisées via un environnement de simulation développé en Python, ont permis de comparer les performances de l'approche hybride avec celles des protocoles traditionnels. Les résultats obtenus mettent en évidence les avantages en termes de consommation d'énergie, de stabilité du routage et de rapidité de convergence, tout en soulignant certaines limites et perspectives d'amélioration futures.

Mots-clés : Internet des Objets, routage, optimisation, communautés, RPL, AODV, Louvain, hybridation, performance, simulation.

Abstract

This thesis focuses on optimizing routing in Internet of Things (IoT) networks, with particular attention to the integration of community-based approaches and hybrid routing strategies. IoT networks, composed of nodes often limited in energy and resources, face numerous challenges such as mobility, scalability, communication reliability, and the management of dynamic constraints.

The study is based on the idea that a community-based organization of nodes can enhance routing efficiency. An original approach is proposed, combining the Louvain community detection algorithm with classical routing protocols—such as RPL (adapted for IoT) for intra-community communication, and AODV (more dynamic) for inter-community communication. Each step of this method has been detailed, implemented, and tested.

Experiments were conducted using a simulation environment developed in Python, allowing for a comparison between the hybrid approach and traditional routing protocols. The results highlight the advantages in terms of energy consumption, routing stability, and convergence speed, while also pointing out some limitations and future improvement perspectives.

Keywords : Internet of Things, routing, optimization, communities, RPL, AODV, Louvain, hybridization, performance, simulation.