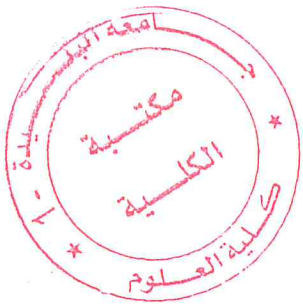
MA-004-478-2

RÉPUBLIQUE ALGERIÈNNE DÉMOCRATIQUE ET POPULAIRE
UNIVERSITÉ SAAD DAHLAB BLIDA1
FACULTÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE



Thème: Conception et implémentation d'une nouvelle métrique de routage pour le protocole RPL pour l'amélioration de la qualité de service.

En vue d'obtenir le diplôme de Master

Domaine: MI

Filière: Informatique

Spécialité: génie des systèmes informatiques

Organisme d'accueil: CERIST
Encadreur: M. BOUCHAMA Nadir
Promoteur: M. OULD-KHAOUA Mohamed

Rapport présenté par:
LAKHDARI Hamida
LARIBI Salah

Année universitaire: 2016-2017

MA-004-478-1

Remerciements

Nous tenons tout d'abord à remercier Dieu le tout puissant, qui nous a donné la force et la patience d'accomplir ce travail. En second lieu, nous tenons à remercier notre encadreur: **M. BOUCHAMA Nadir**, et notre promoteur: **M. OULDKHAOUA Mohamed**, de leurs précieux conseils et leurs aides durant toute la période du travail.

Nos vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils ont porté ànotre recherche en acceptant d'examiner notre travail Et de l'enrichir par leurs propositions. Nos reconnaissances et gratitudes à l'administration et à l'ensemble du corps enseignant de l'Université Saad Dahlab de Blida pour leurs efforts à nous garantir la continuité et laboutissement de ce programme de Master.

Enfin, nous tenons également à remercier toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.

جامعة البليدة ٢
مكتبة
الكلية
كلية العلوم

Table des matières

جامعة البليدة 1 مكتبة الكلية كلية العلوم

Résumé

Le protocole RPL (Routing Protocol for LLNs) [RFC 6550] est un protocole de routage qui a été conçu spécialement par le groupe de travail ROLL (Routing Over LLNs) de l'IETF (Internet Engineering Task Force) pour les réseaux LLNs (Low power and Lossy Networks) ([RFC 5548]. C'est un protocole basé sur IPv6 qui construit un DAG (de l'anglais Directed Acyclic Graph, graphe orienté acyclique) qui a pour racine un sink. Pour améliorer la qualité de service (délai de bout en bout, taux de perte de paquets, débit, etc) pour une application donnée, plusieurs métriques peuvent être combinées dans une fonction objective (OF). L'objectif de ce travail est le suivant:

- Proposer une nouvelle métrique de routage pour le protocole RPL pour améliorer le délai de bout en bout;
- Implémenter le nouveau protocole de routage sur un simulateur;
- Evaluer le nouveau protocole de routage par rapport à RPL.

abstract

The RPL (Routing Protocol for LLNs) protocol [RFC 6550] is a Routing protocol that was specially designed by the ROLL Working Group (Routing Over LLNs) from the Internet Engineering Task Force (IETF) for LLNs (Low Power and Lossy Networks) (RFC 5548). Based on IPv6 which builds a DAG (from the English Directed Acyclic Graph) which has for its root a sink. To improve quality of service (End-to-end delay, packet loss rate, throughput, etc.) for a given application, Several metrics can be combined in an objective function (OF). The objective of this work is as follows:

- Propose a new routing metric for the RPL protocol to improve End-to-end delay;
- Implement the new routing protocol on a simulator;
- Evaluate the new routing protocol in relation to RPL;

Liste des Figures

Liste des Tableaux

Abréviations

6LowWPAN . IPv6 over Low power Wireless Personal Area Networks

AODV ..... Ad-hoc On demand Distance Vector

ALABAMO . A LoAd BAlancing MOdel

A-LLN ..... Agricultural-Low power and Lossy Networks

API ..... Application Programming Interface

CoAP ..... Constrained Application Protocol

CAOF ..... Context-Aware Objective Function

Cyber-OF .... Cyber-Physical objective function

DIS ..... DODAG Information Solicitation

DIO ..... DODAG Information Object

DAO ..... Destination Advertisement Object

DAO-ACK ... Destination Advertisement Object Acknowledgement

DAG ..... Directed Acyclic Graph

D2D ..... Device to Device

DSR ..... Dynamic Source Routing

DTSN ..... Destination Advertisement Trigger Sequence Number

DODAG ..... Destination Oriented Directed Acyclic Graph

D-RPL ..... Delay for RPL

ETX ..... Expected Number of Transmissions

ELT ..... Expected LifeTime

EEPROM .... Electrically-Erasable Programmable Read-Only Memory

FMOF ..... Fuzzy Mobility Objective Function

HC ..... Hop Count

IP ..... Internet Protocol

IETF ..... International Engineering Task Force

IS-IS ..... Intermediate System to Intermediate System

IoT ..... Internet of Things

IPv6 ..... Internet Protocol version 6

IEEE ..... Institute of Electrical and Electronics Engineers

ICMPv6 ..... Internet Control Message Protocol version 6

LLN ..... Low power and Lossy Networks

LR-WPAN ... Low Rate-Wireless Personal Area Network

LQL ..... Link Quality Level

LPWNs ..... Low-Power Wireless Networks

MP2P ..... Multipoint-to-Point

MOP ..... Mode of Operation

MAC ..... Medium Access Control

MRHOF ..... Minimum Rank with Hysteresis Objective Function

ND ..... Neighbor Discovery

OCP ..... Objective Code Point

OSPF ..... Open Shortest Path First

OLSR ..... Optimized Link State Routing Protocol

OF0 ..... Objective Function0

OF-FL ..... Objective Function based on Fuzzy Logic

PFI ..... packet transfer indicator

Prf ..... DODAGPreference

P2MP ..... Point-to-Multipoint

P2P ..... Point-to-point

PDR ..... Packet Delivery Ratio

QoS ..... Qualite of Service

RFID ..... Radio Frequency Identification

RPL ..... Routing Protocol for Low Power and Lossy Networks

RREQ ..... Route Request

RREP ..... Route Reply

RERR ..... Route Error

ROLL-WG ... ROLL Working Group

RPAL ..... Agricultural Low-Power and Lossy Networks

RE ..... Energie restante

RSSI ..... Received Signal Strength Indicator

RX ..... Receive

SCAOF ..... Scalable Context-Aware Objective Function

TCP/IP ..... Transmission Control Protocol/Internet Protocol

TBRPF ..... Topology dissemination Based on Reverse Path Forwarding

TX ..... Transmit

UDGM ..... Unit Disk Graph Model

UDP ..... User Datagram Protocol

WSN ..... Wireless Sensor Networks

WPAN ..... Wireless Personal Area Network

ZRP ..... Zone based hierarchical link stateRouting Protocol

Introduction générale

Recemment avec la création de réseaux de capteurs sans fil et leur intégration avec Internet, l'Internet des objets devient rapidement une réalité, cette denière comporte un grand nombre d'objets physiques connectés entre eux et qu'ils sont capable d'accéder à l'internet. Ces objets sont généralement munis de capteurs (capteurs de température, de vibrations, de luminosité, etc.) et/ou d'actionneurs afin de surveiller ou d'interagir avec leur environnement.

RPL a été conçu parce qu'aucun des protocoles connus existants ne satisfaisait aux exigences spécifiques de réseaux de capteurs sans fil à grande échelle et à faible puissance. Le protocole RPL cible les réseaux de capteurs sans fil à grande echelle et supporte une variété d'applications. Le protocole de routage conçu devrait fonctionner sur une variété d'application, y compris les réseaux de capteurs sans fil à grande échelle et à faible puissance, mais sans s'y limiter. Cette fonctionnalité nécessite le protocole RPL pour prendre en charge l'hétérogénéité dans ces réseaux.

la problématique est que RPL ne répond pas aux scénarios d'urgence où il existe un trafic de données élevé dans le réseau. En cas d'urgence, un certain nombre de messages peuvent etre envoyés provoquant une congestion dans le réseau. Cela provoque également des retards et une éventuelle perte de paquets [68]. RPL ne spécifie aucun mécanisme pour faire face à la perte de paquets de données. En outre, les nœuds sont contraints à des ressources, de sorte que même leur utilisation pour attendre que le réseau soit décongestionné ne soit pas possible. Les chemins établis par le protocole RPL standard ne permettent pas de garantir des critères de qualité de service. C'est pourquoi il semble important de faire une extension

de ce protocole afin d'assurer une certaine qualité de service quit peut être integré dans RPL par l'introduction de la métrique "délai" qui est plus appropriée que la distance (nombre de sauts) et qui permet de choisir le chemin le moins congestionné.

Dans ce mémoire nous allons organiser nos chapitres comme suit :

Chapitre 1: donne un état de l'art sur l'internet des objets et ses différents concepts et précise les notions de base de l'internet des objets et ses principales caractéristiques ainsi que les contraintes qui en découlent. Il décrit également les deux protocoles RPL et LOAD et leurs classifications.

Chapitre 2: donne une vue sur la notion de la QoS et ses défis, la motivation pour le protocole RPL , et sur quelques fonctions objectives existantes.

Chapitre 3: est consacré à la conception du protocole RPL en termes du délai, il décrit egalement la méthode d'estimation du délai utilisée.

Chapitre 4: donne des détails de l'implémentation de ce qui a été conçu.

Chapitre 5: est consacré aux simulations et discussions des résultats.

Nous terminerons par une conclusion générale et quelques perspectives pour des travaux futurs.

Chapitre 1 : Etat de l'art sur l'Internet des Objets

Introduction

On assiste récemment à l'émergence de l'Internet des objets, qui est un nouveau paradigme bouleversant le domaine des réseaux de télécommunication. L'Internet des objets qui est une partie intégrante dans l'Internet du futur, consiste en une large interconnexion de toutes sortes d'objets (autre que les ordinateurs et les téléphones mobiles) dans notre entourage.

Dans ce chapitre nous présentant un état de l'art sur l'Internet des objets en donnant une présentation de l'Internet des objets et tout ce qui s'y rapporte, ainsi qu'une étude détaillée sur le protocole RPL

1.1 Internet des objets

L'internet des objets (ou IoT) se traduit à l'heure actuelle par l'accroissement du nombre d'objets connectés, c'est-à-dire d'appareils possédant une identité propre et des capacités de calcul et de communication de plus en plus sophistiquées : téléphones, capteurs, appareils ménagers, etc.

Ces objets embarquent un nombre grandissant de capteurs et d'actionneurs leur permettant de mesurer l'environnement et d'agir sur celui-ci, faisant ainsi le lien entre le monde physique et le monde virtuel.

- Le terme Internet des Objets a été inventé par Kevin Ashton en 1999 dans le cadre de la gestion de la chaîne d'approvisionnement [2]. Qui ve dire un "réseau mondial d'objets interconnectés exclusivement adressables, basés sur des protocoles de communication

standards" [10].

Cependant, au cours de la dernière décennie, la définition a été plus inclusive couvrant une large gamme d'applications comme les transports, les soins de santé, les services publics, etc. [20]. Bien que la définition des choses ait changé au fur et a mesure que la technologie évoluait, le but principal de rendre l'information informatique sans l'aide de l'intervention humaine reste la même. Une évolution radicale de l'Internet actuel en un réseau d'objets interconnectés qui non seulement récolte des informations provenant de l'environnement (détection) et interagit avec le monde physique (action/commande/contrôle), mais utilise également les normes Internet existantes pour fournir des services de transfert d'informations , L'analyse, les applications et les communications.

- Selon le groupe de projets de recherche européens sur l'Internet des objets [20] :

  Les objets sont des participants actifs capables d'interagir et de communiquer entre eux et avec l'environnement en échangeant des données et des informations sensées sur l'environnement tout en réagissant de manière autonome aux événements mondiaux/réels En exécutant des processus qui déclenchent des actions et créent des services avec ou sans intervention humaine directe.

- Le groupe RFID définit l'Internet des objets comme le réseau mondial d'objets interconnectés uniquement adressables basés sur des protocoles de communication standard.

1.1.1 Eléments principales d'IoT

D'un point de vue technique, l'Internet des objets n'est pas le résultat d'une seule technologie, au lieu de cela, plusieurs développements techniques complémentaires fournissent des capacités qui, prises ensemble, contribuent à combler le fossé entre le virtuel monde et le monde physique. Ces capacités comprennent:

- **Radio Frequency Identification (RFID)**

  La technologie RFID est une percée majeure dans le paradigme de la communication

intégrée qui permet la conception de micro-puces pour la communication de données sans fil. Ils aident à l'identification automatique de tout ce qu'ils sont attachés avec, agissant comme un code-barres électronique [24] Un dispositif RFID - fréquemment appelésimplement étiquette RFID est une petite puce conçue pour la transmission de données sans fil, qui est généralement fixé à une antenne dans un emballage qui ressemble un autocollant adhésif ordinaire. La micropuce elle-même peut être petite comme un grain de sable[21].

- **Les réseaux de capteurs**

  Les progrès technologiques récents dans les circuits intégrés de faible puissance et les communications sans fil ont rendu disponibles des dispositifs miniatures efficaces, à faible coût et à faible puissance pour utilisation dans des applications de télédétection. La combinaison de ces facteurs a permis d'améliorer la viabilité de l'utilisation d'un réseau de capteurs composé d'un grand nombre de capteurs intelligents permettant la collecte, le traitement, l'analyse et la diffusion d'informations précieuses recueillies dans divers environnements.

  Les données des capteurs sont partagées entre les nœuds des capteurs et envoyées à un système distribué ou centralisé pour l'analyse.

  Les nœuds devraient être déployés de manière Ad hoc pour la plupart des applications, et ils doivent communiquer entre eux pour transmettre des données en simple ou multi-sauts à une station de base. La pile de communication au nœud de réception (sink) devrait pouvoir interagir avec le monde extérieur par le biais d'Internet pour agir comme une passerelle vers le sous-réseau WSN et l'Internet [7].

  La position des nœuds de capteurs ne doit pas être prédéterminée. Cela permet un déploiement aléatoire dans des terrains inaccessibles ou des opérations de secours en cas de catastrophe. D'autre part, cela signifie que les protocoles de réseau de capteurs

et les algorithmes doivent posséder des capacités d'auto-organisation [1].

Une des contraintes les plus importantes sur les nœuds de capteurs est l'exigence de faible consommation d'énergie. Les nœuds de capteurs portent des fonctions limitées, généralement des sources d'énergie irremplaçables. Alors que les réseaux traditionnels visent à atteindre une qualité de service élevée (QoS).

Les protocoles du réseau de capteurs doivent concentrer principalement sur la conservation de l'énergie. Ils doivent avoir des mécanismes de compromis qui permettent à l'utilisateur final de prolonger la durée de vie du réseau au coût de débit inférieur ou de retard de transmission plus élevée.

- **Stockage et analyse des données**

  Les nœuds de capteurs sont équipés d'un processeur embarqué ou d'un microcontrôleur ainsi qu'une capacité de stockage. Ces ressources peuvent être utilisées, par exemple, pour effectuer localement des calculs simples et transmettre uniquement les données requises et partiellement traitées Au lieu d'envoyer les données brutes aux nœuds responsables de la fusion.

- **Interfaces d'utilisateur**

  La visualisation est essentielle pour une application IoT car cela permet l'interaction de l'utilisateur avec l'environnement. Les objets intelligents peuvent communiquer avec les personnes de manière appropriée (directement ou indirectement, par exemple via un smartphone). Avec les récentes avancées dans les technologies d'écran tactile, l'utilisation de tablettes et de téléphones intelligents est devenue très intuitive.

- **Systèmes d'adressage**

  La possibilité d'identifier les objets d'une manière unique est essentielle pour le succès de l'IoT. Cela nous permettra non seulement d'identifier de façon unique des milliards de périphériques, mais aussi de contrôler des périphériques distants via Internet. Les caractéristiques les plus critiques de la création d'une adresse unique sont: l'unicité, la

fiabilité, la persistance et l'évolutivité [9]. La RFID et les codes à barres lisibles optiquement sont des exemples de technologies avec lesquels les objets peuvent être identifiés (à l'aide d'un médiateur tel qu'un lecteur RFID ou un téléphone). L'identification permet d'associer des objets à des informations associées à l'objet particulier et qui peuvent être récupérées à partir d'un serveur.

1.1.2 Applications d'IoT

Il existe de nombreuses applications de l'IoT dans l'automatisation industrielle, la surveillance et le contrôle du trafic, la surveillance des dispositifs médicaux et dans d'autres domaines.

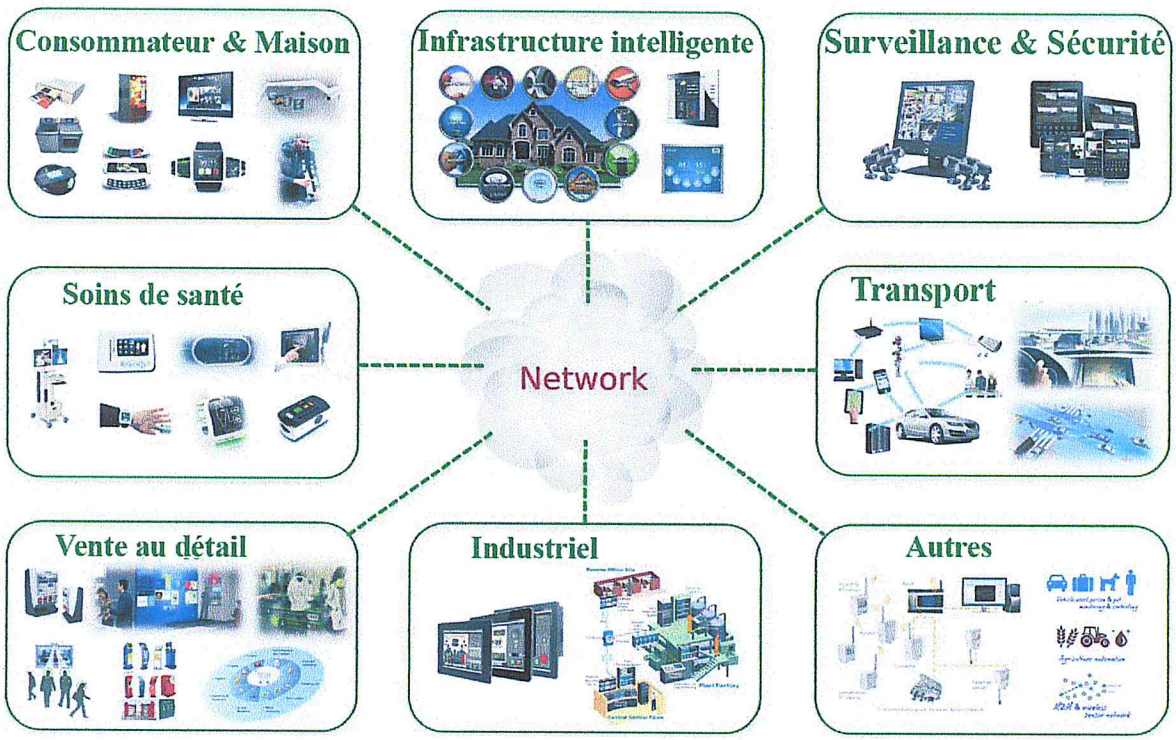


Figure 1.1: Applications d'IoT

Les réseaux de capteurs sans fil (WSN) ont gagnés une attention croissante dans la détection de la circulation et en évitant la congestion routière. Les WSNs sont très utilisés en

raison de leur transfert plus rapide de l'information, l'installation facile, moins d'entretien, la compacité et d'être moins cher par rapport à d'autres options du réseau.

De nombreux systèmes de feux de circulation fonctionnent sur un mécanisme de synchronisation qui change les lumières après un intervalle donné. Un système intelligent de feux de circulation détecte la présence ou l'absence de véhicules et réagit en conséquence. L'idée derrière les systèmes de trafic intelligent est que les conducteurs ne passeront pas le temps inutile en attendant que les feux de circulation changent. Un système de trafic intelligent détecte le trafic à l'aide de WSN, RFID, ZigBee, appareils Bluetooth, caméras et signaux infrarouges.

Les réseaux de capteurs utilisés dans le contrôle du trafic se composent d'un capteur et d'un nœud de passerelle. Le rôle du nœud de capteur est de surveiller dans une zone allouée, en utilisant différents dispositifs qui peuvent mesurer plusieurs paramètres physiques du trafic comme la circulation, la densité, le volume, l'avancée, le temps d'attente, aussi bien que la pollution. Le nœud de passerelle collecte les informations de trafic de tous les nœuds et les dirige vers la station de base. IoT ainsi que d'autres technologies sont censés créer une révolution dans la gestion du trafic et l'amélioration des systèmes de contrôle.

1.2 Routage dans IoT

Selon **Oladayo Bello et al.** un protocole de routage intelligent peut libérer la puissance intrinsèque de tout réseau hétérogène, dynamique et complexe qui est caractérisé par de multiples facteurs dynamiques tels que le changement de topologie et de flux. Ainsi pour obtenir la pleine fonctionnalité de l'IoT, des protocoles intelligents sont nécessaires pour la communication D2D (Device to Device) dans IoT. Des protocoles de routage efficaces et evolutifs adaptables à différents scénarios et variations de réseaux,

càpable de trouver des routes optimales sont nécessaires.

1.2.1 Pile protocolaire d'IoT

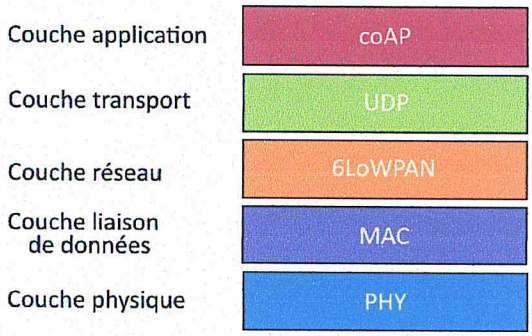


Figure 2.1: Pile protocolaire d'IoT

Comme le montre la figure 2.1, la pile protocolaire de l'IoT est constituée des couches suivantes:

Couche application

Bien que les protocoles web sont disponibles et utilisables pour les périphériques IoT, ils sont trop lourds pour la majorité des applications IoT. Le protocole CoAP (The Constrained Application Protocol) a été conçu par l'IETF pour une utilisation avec des réseaux à faible puissance et contraints, et il est un bon choix pour les appareils fonctionnant sur des batteries ou la récolte d'énergie.

Couche transport

Avec UDP (User Datagram Protocol), les applications peuvent envoyer des messages, dans ce cas, appelés datagrammes, vers d'autres hôtes sur un réseau IP sans communication préalable pour configurer des canaux de transmission spéciaux ou des chemins de données. Les data-

grammes UDP sont si simples, l'en-tête contient le port et l'adresse source, le port et l'adresse de destination, une longueur et une somme de contrôle.

Couche réseau

6LowPAN est un protocole réseau qui définit des mécanismes d'encapsulation et de compression d'en-tête qui permettent d'envoyer et de recevoir des paquets IPv6 sur des réseaux locaux sans fil (LR-WPAN) à faible débit.

Couche liaison de données

La couche MAC (The Medium Access Control) permet la transmission de trames MAC à l'aide de la couche physique. Cette couche offre aussi une interface de gestion et gère lui-même l'accès à la couche physique. Elle contrôle également la validation des trames, garantit les créneaux temporels et gère les associations de nœuds. Enfin, Elle offre des points d'accrochage pour des services sécurisés.

Couche physique

La couche physique fournit le service de transmission de données, ainsi que l'interface avec l'entité de gestion de couche physique, qui offre l'accès à chaque fonction de gestion de couche et maintient une base de données d'informations sur des réseaux personnels associés.

1.2.2 Types de protocoles de routage

Les protocoles de routage sont classés en des protocoles proactif, réactif et hybride en ce qui concerne la manière dont ils prennent les décisions de routage.

- **Les protocoles proactifs** maintiennent toujours l'information de chemin sous forme de tableau à tout moment. Les protocoles proactifs sont utilisés dans les réseaux statiques où la topologie est fixée la plupart du temps. Comme le routage est toujours basé sur une table de routage, ces protocoles sont également appelés "protocoles basés sur table".

- **Les protocoles réactifs** ne maintiennent pas l'information de l'itinéraire, les itinéraires sont formés comme et quand ils sont requis. Les protocoles réactifs sont utilisés dans les réseaux dynamiques où il existe des changements topologiques fréquents. Comme IoT supporte les topologies dynamiques la plupart du temps, du point de vue du chercheur, les protocoles réactifs revêtent une importance particulière.
- **Le routage hybride** fait usage des algorithmes de routage proactif et réactif .

Type de protocole 1	**Exemples**
Proactif	Optimized linked state routing (OLSR), Topology dissemination Based on Reverse Path Forwarding (TBRPF),Routing Protocol for Low Power and Lossy Networks(RPL)
Réactif	Dynamic source routing (DSR), Ad-hoc on demand distance vector (AODV), LOAD
Hybrid	Zone based hierarchical link state routing protocol (ZRP)

Table 2.1: Types des protocoles de routage

Cette classification peut être étendue en ajoutant d'autres paramètres tels que la qualité de service (QoS). Aujourd'hui, les exigences changent et les chercheurs sont engagés dans la conception d'algorithmes de routage plus intelligents qui comprendront l'environnement et l'état exact du réseau.

1.2.3 Comparaison entre routage distribué et centralisé

Dans une approche centralisée, il y a un nœud racine qui est supposé d'avoir des ressources abondantes et des connaissances sur l'état de tous le réseau. Ce nœud racine contrôle tous

les autres nœuds, calcule le chemin de routage optimal pour chaque paquet de données et adapte les chemins de routage en conséquence. L'avantage du routage centralisé est le contrôle complet sur tous les aspects du réseau, donc des chemins de routage optimum pourraient être calculés.

Dans une approche distribuée, un nœud individuel ou un ensemble de nœuds qui sont à proximité les uns des autres prennent la décision de routage. Ces nœuds ne connaissent pas l'état de tout le réseau, mais seulement leur état local (et éventuellement l'état de leurs voisins). La décision d'acheminement, par conséquent, est faite seulement en fonction de cette connaissance limitée. Les avantages du routage distribué sont la flexibilité à mesure que la prise de décision est distribuée et réalisée par chaque nœud, et la réactivité parce que les nœuds à proximité peuvent réagir rapidement à tout problème lié à la dynamique qui se produit localement.

Les inconvénients sont peut-être les chemins de routage sous-optimale et la répartition de la charge potentiellement déséquilibrée, puisque seules les informations locales sont utilisées. Presque tous les protocoles de routage conçus pour l'IoT sont distribués pour assurer l'évolutivité[17].

1.2.4 Techniques d'optimisation du routage

1. Acheminement efficace: Optimise les besoins en énergie tout en sélectionnant un chemin vers la destination qui contribue à augmenter la durée de vie du réseau. Cette technique achemine les données vers la destination par l'intermédiaire des nœuds qui ont des ressources énergétiques suffisantes et évite la participation de nœuds ayant une énergie inférieure à une valeur de seuil spécifique.

2. Elimination de la redondance des données: Plus de données signifie plus d'énergie requise pour le routage. La plupart du temps il y a redondance dans les données. L'élimination de la redondance réduira l'exigence énergétique pour le routage des données, ce qui entraînera

une augmentation de la durée de vie du réseau. Cela génère un besoin de développer des techniques de fusion de données.

3. Minimisation du retard: IoT contient une population dense des dispositifs générant une énorme quantité de données. L'expiration est associée aux données. De ce fait, il est essentiel d'envoyer les données à la destination dans un laps de temps fixe. En raison de cela, la minimisation du délai est nécessaire.

1.3 Problèmes de routage dans IoT

Comme IoT contient un ensemble de composants mobiles et stationnaires, de multiples problèmes se posent dans le développement de protocoles de routage où ces périphériques inter communiquant entre eux. Et comme plusieurs facteurs sont dominants dans le fonctionnement du protocole de routage, il devient donc difficile de concevoir un protocole unique qui atteindra tous ces objectifs qui sont par nature paradoxaux.

1. **Ressources limitées:**

   L'un des principaux défis de l'IoT est la limitation des ressources, y compris l'approvisionnement en énergie, la puissance de traitement, les capacités de mémoire, la portée de communication sans fil et la bande passante de communication sans fil. Cette limitation affecte
   le routage de plusieurs façons. La plage de communication sans fil courte exige que le routage doit être effectué d'une manière multi-mode, c'est-à-dire que les paquets de données doivent être acheminés par plusieurs nœuds de relais afin d'atteindre leur des tination. La faible puissance de traitement et la mémoire de programme exigent que le processus de routage en cours sur les périphériques IoT doit être hautement optimisé et léger. La faible mémoire de stockage et la bande passante de communication peuvent limiter la taille des paquets à transmettre. La rareté de la source d'énergie (fournie par la batterie ou récoltée) rend difficile la décision des nœuds qui doivent transmettre les paquets de données, car la communication sans fil domine la consommation d'énergie des dispositifs IoT.

2. **Topologie de routage dynamique:** La cause de la dynamique de la topologie de routage est:

   - en raison de la contrainte d'énergie, les dispositifs IoT sont généralement programmés pour être inactifs ou fonctionnant (par exemple, en activant / désactivant la radio sans fil) pour minimiser la consommation d'énergie, rendant la topologie de routage dynamique.
   - puisque les utilisateurs déploient ou retirent leurs dispositifs à volonté, les nœuds de seront connectés et déconnectés de réseau à une vitesse inconnue, ce qui ajoute l'imprévisibilité à la dynamique de la topologie de routage.
   - les échecs de nœuds sont fréquents dans l'IoT. Les causes d'une défaillance comprennent un dysfonctionnement du matériel (p. Ex., Endommagement de l'antenne), un approvisionnement en énergie épuisé (par exemple, des piles appauvries) et un impact environnemental.
   - la mobilité des nœuds provoque la reconfiguration des liaisons sans fil entre les nœuds mobiles et les autres nœuds à proximité.
   - les liaisons sans fil de faible puissance dans les réseaux de dispositifs IoT (par exemple, WPAN, WSN) sont peu fiables et transitoires, ce qui contribue également a la dynamique de la topologie. Les protocoles de routage doivent donc être suffisamment souples pour faire face à une telle dynamique de la topologie de l'IoT.

3. **´evolutivité:** L'IoT sera de grande échelle, tant en termes de nombre de nœuds que géographiquement.

   Comme le routage veut dire de décider sur quel chemin le paquet de données doit être envoyé, plus les nœuds de relais candidats à évaluer pour l'inclusion dans un chemin de routage, le routage devient plus complexe. Cette complexité est multiple, y compris la fonction de coût à utiliser, comment décider lequel des voisins d'un nœud est le nœud relais, quel est le coût d'installation et de maintenance d'un chemin de routage, comment installer un nouveau chemin de routage quand un autre est cassée, etc.

Une telle complexité deviendra rapidement impossible à gérer si le protocole de routage n'a pas été soigneusement conçu et que le défi d'évolutivité a été pris en compte. En d'autres termes, le protocole doit être évolutif pour tout réseau dont la taille fluctue de 100 à 1 000000 de nœuds, tant en termes de performances que d'utilisation de la mémoire.

4. **Partitions et vides:** Un autre défi majeur au routage dans l'IoT est la présence de partitions de réseau et de vides dans le réseau. Une partition est une partie déconnectée du réseau, de sorte que les nœuds à l'intérieur d'une partition ne peuvent pas communiquer avec les nœuds dans les autres parties du réseau, car il n'y a pas de chemin d'acheminement pour échanger des paquets de données. Un vide est une zone qui n'est pas couverte par le réseau. Puisqu'il n'y a aucun nœud situé à l'intérieur du vide qui est connecté au nœud(s) à l'extérieur de celui-ci, les paquets de données ne peuvent être transmis autour du vide que pour atteindre leur destination. Par exemple, un WSN á eté déployé en dispersant de manière aléatoire un grand nombre de nœuds de capteurs sur une zone géographique. En raison de la structure de la zone, il peut y avoir des lacs qui causent des vides, ou des rivières qui provoquent des partitions dans le WSN.

5. **Utilisation du transmetteur:** En ce qui concerne la consommation d'énergie, l'émetteur est le composant le plus cher d'un dispositif contraint. Il est donc conseillé de l'utiliser aussi peu que possible.

1.4 Quelques protocoles de routage

Cette section décrit certains protocoles de routage récents dans IoT.

1.4.1 LOAD

LOAD [12] est un protocole, dérivé de AODV [15] et adapté pour LLNs. Ainsi, le fonctionnement de base de LOAD est identique à celui de l'AODV :
Un périphérique avec un paquet à livrer à une destination et qui n'a pas d'entrée valide dans

sa table de routage pour cette destination, émet un message de demande d'itinéraire (RREQ) diffusé par le réseau de façon à atteindre tous les autres dispositifs.

Lorsqu'un périphérique transmet cette demande d'itinéraire, il enregistre une entrée dans sa table de routage vers l'expéditeur de cette demande (une route inverse indiquant le chemin éventuel de la destination à l'expéditeur). Si la destination est présente dans le réseau, elle recevra éventuellement la demande d'itinéraire et répondra par une réponse d'itinéraire (RREP), unicast à l'expéditeur de demande d'itinéraire le long de l'itinéraire inverse précédemment installé.

Comme cette réponse d'itinéraire est envoyée le long de cette route inverse, Les dispositifs qui l'envoient va instiller une route vers la destination. Une fois que la réponse d'itinéraire arrive à l'initiateur de la demande d'itinéraire correspondant, un chemin bidirectionnel est installé, disponible pour l'utilisation.

Opération de réparation d'itinéraire: Lorsqu'un lien est détecté comme étant brisé, le routeur de détection peut s'engager dans une opération de réparation d'itinéraire essentiellement un nouveau cycle de demande/réponse d'itinéraire pour découvrir un chemin vers la destination et si cela échoue, envoyer un message d'erreur d'itinéraire (RERR) pour informer la source de données défaillant de l'erreur.
Bien que cette découverte d'itinéraire soit effectuée, tous les paquets IP à la destination sont mis en mémoire tampon dans le routeur source. Quand un itinéraire est établi, ces paquets sont transmis et si aucun itinéraire ne peut être établi, ils sont abandonnés.

Comparaison entre AODV et LOAD: Les principales différences entre AODV et LOAD sont les suivantes:
1. LOAD simplifie le comportement du protocole en refusant que les dispositifs intermédiaires répondent par une réponse d'itinéraire même s'ils ont une route active vers la destination prévue, éliminant ainsi le besoin de numéros de séquence de destination.
2. Dans le cas d'un AODV, dans le cas où un dispositif détecte une rupture de liaison, ce

dispositif tentera de transmettre le message d'erreur de route à tous les voisins qui l'ont récemment utilisé comme un prochain saut sur un chemin vers la destination du paquet non délivré, LOAD désactive cette transmission éliminant le besoin d'un dispositif pour maintenir une liste de précurseurs.
D'autres différences mineures comprennent la simplification du format de paquet, prise en charge des adresses IPv6 compressées [13] etc. LOAD n'impose aucun rôle spécifique sur des dispositifs spécifiques, n'a notamment aucun contrôleur ou racine avec des responsabilités spécifiques pour le fonctionnement du réseau. Ainsi, Le modèle de trafic par défaut pris en charge soutenu par LOAD est bi-directionnel point-à-point. Le seul sacrifice que LOAD fait avec en ce qui concerne le trafic de données, la simplification de l'AODV est que LOAD suppose qu'une destination donnée est typiquement en communication avec une seule source à un moment donné, dès la suppression de la liste des précurseurs.

1.4.2 RPL

Protocole de routage sur les réseaux à faible puissance et à perte (RPL).

Le protocole de routage pour les réseaux à faible consommation et à perte (RPL) est spécifiquement conçu pour les réseaux de capteurs sans fil (WSN) dans l'Internet des objets. Il fonctionne sur la couche IPv6 6LoWPAN au-dessus 802.15.4 standard de réseau sans fil. Le protocole est divisé en un moteur de transfert qui utilise une table de routage pour décider quel nœud voisin est le prochain saut pour un paquet donné et un protocole de routage qui remplit une table de routage à chaque nœud. Le protocole de routage fonctionne en attribuant à chaque nœud un rang, de sorte que le rang d'un nœud augmente plus le nœud est distant du routeur de bordure. Cela crée un graphe des chemins de communication à travers le réseau appelé un graphique acyclique dirigé orienté vers la destination (DODAG) [1].

Un réseau peut consister en un ou plusieurs DODAG, qui forment ensemble une instance RPL identifiée par un identifiant unique, appelé RPLInstanceID. Un réseau peut exécuter simultanément plusieurs instances RPL, Mais ces instances sont logiquement indépendantes. Un nœud peut joindre plusieurs instances RPL, mais doit appartenir à un seul DODAG dans

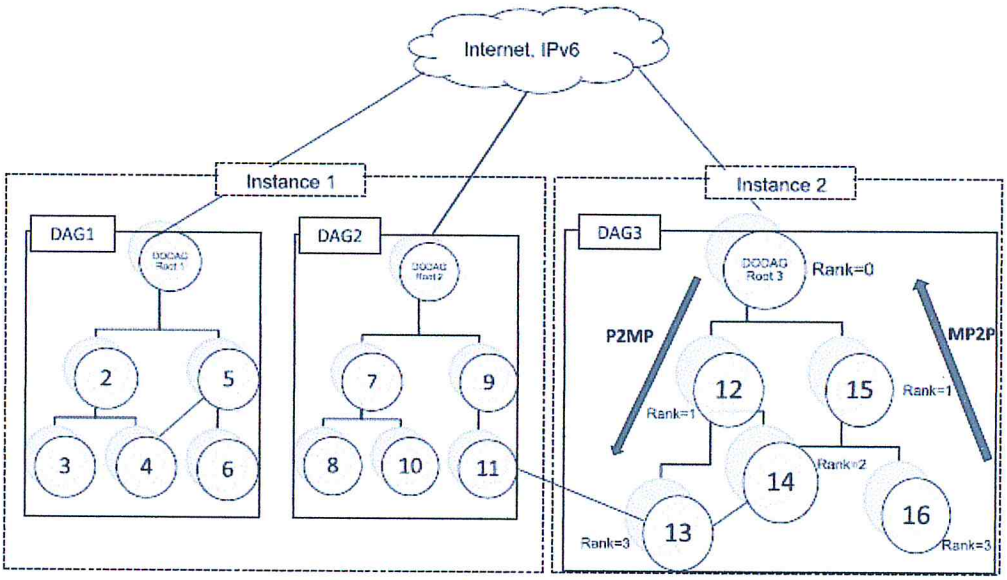


Figure 2.2: Un réseau RPL avec trois DODAG dans deux instances

chaque instance. La figure 1 montre un exemple d'instances RPL avec plusieurs DODAG.

Pour éviter et détecter les boucles de routage, et permet aux nœuds de faire la distinction entre leurs parents et leurs frères dans le DODAG, chaque nœud du DODAG reçoit un rang. Le rang d'un nœud est défini dans [25] comme la position individuelle du nœud par rapport à d'autres nœuds par rapport à une racine DODAG. C'est un entier qui représente l'emplacement d'un nœud dans le DODAG. Les nœuds au-dessus de la hiérarchie reçoivent des rangs plus petits que ceux dans le fond et le plus petit rang est assigné à la racine de DODAG. C'est illustré sur la figure 2.2.

Les fonctions objectives

La fonction Objective définit comment les nœuds RPL traduisent une ou plusieurs métriques en rangs, et comment sélectionner et optimiser des itinéraires dans un DODAG. Elle est responsable du calcul du classement basé sur des mesures de routage spécifiques (par exemple, délai, qualité de la liaison, connectivité, etc.) et spécifie les contraintes de routage et les objectifs d'optimisation.

Messages de contrôle de RPL

Les messages RPL sont spécifiés comme un nouveau type de messages de contrôle ICMPv6, dont la structure est décrite à la figure 2.3.
Selon [4], le message de contrôle RPL est composé de (i) un en-tête ICMPv6, qui comprend trois champs: Type, Code et Checksum, (ii) un corps de message comprenant une base de messages et un certain nombre d'options.

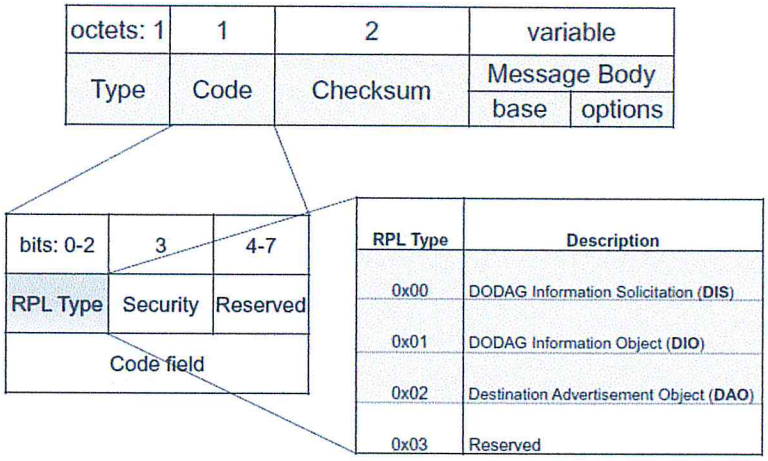


Figure 2.3: Format de message de controle RPL

Le champ Type spécifie le type des messages de contrôle ICMPv6 défini prospectivement à 155 en cas de RPL. Le champ Code identifie le type des messages de contrôle RPL. Quatre codes sont actuellement définis:

DODAG Information Solicitation (DIS): Le message DIS est utilisé pour solliciter un objet d'information DODAG (DIO) à partir d'un nœud RPL. Le DIS peut être utilisé pour sonder les nœuds voisins dans les DODAG adjacents. Le format de message DIS actuel contient des indicateurs et des champs non spécifiés pour une utilisation future.

DODAG Information Object (DIO): Est émise par la racine de DODAG pour construire un nouveau DAG et ensuite envoyée en multidiffusion à travers la structure DODAG.

Ce message sera reçu par un nœud prêt à rejoindre ou un nœud déjà joint. Il annonce à d'autres nœuds "s'ils sont intéressés à se joindre, faites-moi savoir".
Le format de l'objet de base DIO est présenté dans la figure 2.4. Les principaux champs de l'objet de base DIO sont:

RPLInstanceID: Est une information de 8 bits initiée par la racine DODAG qui indique l'ID de l'instance RPL dont le DODAG fait partie.

Grounded (G): Est un flag indiquant si le DODAG actuel satisfait à l'objectif défini par l'application. Si le flag est positionné, le DODAG est mis à la terre. Si le flag est effacé, le DODAG est flottant.

Mode of Operation (MOP): Identifie le mode de fonctionnement de l'instance RPL définie par la racine DODAG.

DODAGPreference (Prf): Un entier non signé de 3 bits qui définit comment la racine de ce DODAG est préférable à d'autres racines DODAG dans l'instance. DAGPreference varie de 0x00 (le moins préféré) à 0x07 (le plus préféré). La valeur par défaut est 0 (le moins préféré).

Version Number: indique le numéro de version d'un DODAG qui est généralement incrémenté à chaque mise à jour d'informations réseau et aide à maintenir tous les nœuds synchronisés avec les nouvelles mises à jour.

Destination Advertisement Trigger Sequence Number (DTSN): est un indicateur de 8 bits qui est utilisé pour maintenir des itinéraires descendants.

Rank: un champ de 16 bits qui spécifie le rang du nœud envoyant le message DIO.

Flags: Champ non utilisé de 8 bits réservé aux flags. Le champ doit être initialisé à zéro par l'expéditeur et doit être ignoré par le récepteur.

Reserved: Champ non utilisé de 8 bits. Le champ doit être initialisé à zéro par l'expéditeur et DOIT être ignoré par le récepteur.

DODAGID: Une adresse IPv6 128 bits définie par une racine DODAG qui identifie de manière unique un DODAG. Le DODAGID doit être une adresse routable IPv6 appartenant à la racine DODAG.

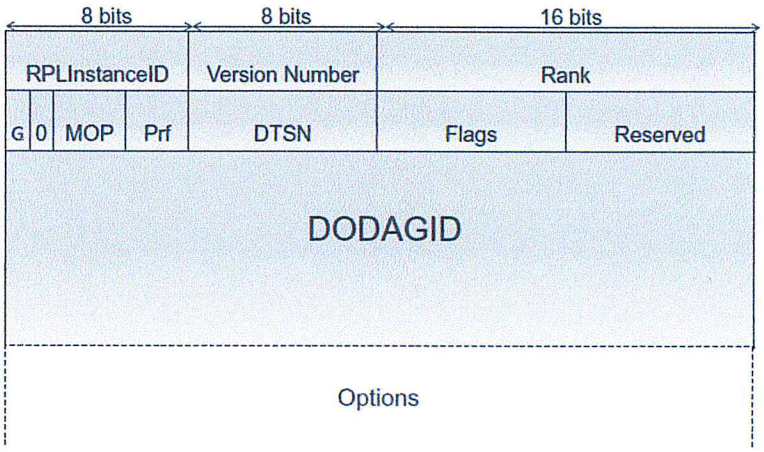


Figure 2.4: Format de message DIO

Destination Advertisement Object (DAO): Il s'agit d'une demande envoyée par un fils à un parent, ce message demande à permettre à un fils de rejoindre le DODAG.

Les messages DAO sont envoyés par chaque nœud, autre que la racine de DODAG, pour remplir les tables de routage avec les préfixes de leurs enfants et pour annoncer leurs adresses et préfixes à leurs parents. Après avoir passé ce message DAO à travers le chemin d'un nœud particulier à la racine DODAG à travers les routes DAG par défaut. En mode Stockage, le message DAO est unicast par l'enfant au parent(s) sélectionné(s). En mode Non-stockage, le message DAO est unicast à la racine DODAG. Le format de l'objet de base DAO est présenté

dans la figure 2.5:

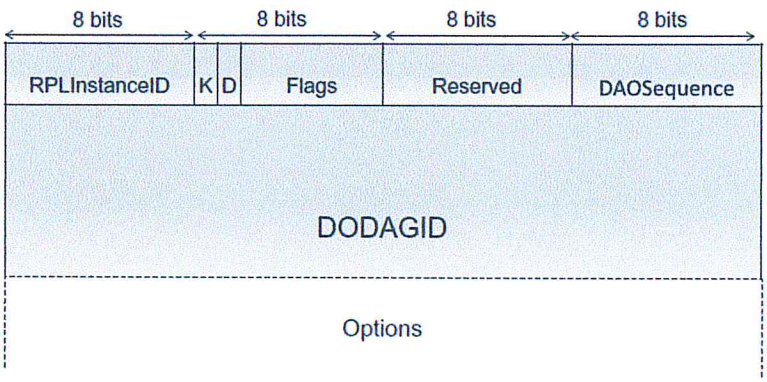


Figure 2.5: Format de message DAO

RPLInstanceID: Un champ de 8 bits indiquant l'instance de topologie associée au DODAG, comme l'a appris le DIO.

K: Indique que le destinataire doit retourner un DAO-ACK.

D:Indique que le champ DODAGID est présent. Ce drapeau DOIT être défini quand un RPLInstanceID local est utilisé.

Flags: Les 6 bits restant inutilisés dans le champ Flags sont réservés aux flags. Le champ doit être initialisé à zéro par l'expéditeur et doit être ignoré par le récepteur.

Reserved: Un champ non utilisé de 8 bits. Le champ doit être initialisé à zéro par l'expéditeur et doit être ignoré par le récepteur.

DAOSequence:incrémenté à chaque message DAO unique à partir d'un nœud et écho dans le message DAO-ACK.

DODAGID (facultatif): Un entier non signé de 128 bits défini par une racine DODAG qui identifie de manière unique un DODAG. Ce champ n'est présent que lorsque l'indicateur

'D' est activé. Ce champ est généralement seul présent lorsqu'un RPLInstanceID local est en cours d'utilisation, afin d'identifier le DODAGID qui est associé au RPLInstanceID. Lorsqu'un ID RPLInstance global est utilisé, ce champ ne doit pas nécessairement être présent.

Destination Advertisement Object Acknowledgement (DAO-ACK): Le message DAO-ACK est envoyé en tant que paquet monodiffusion par un destinataire DAO (un DAO parent ou une racine DODAG) en réponse à un message DAO. Le format de l'objet de base DAO-ACK est présenté dans la figure 2.6:

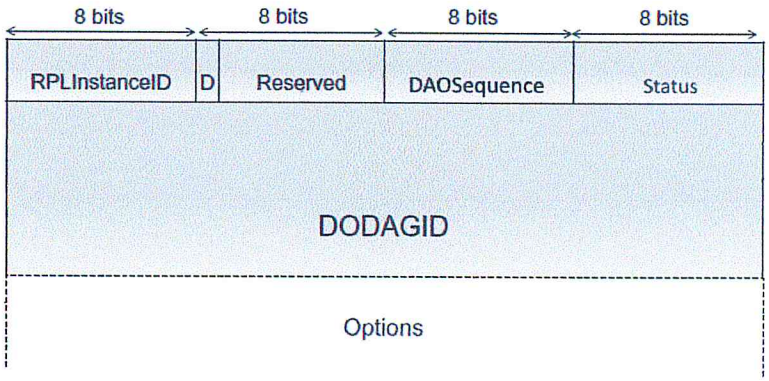


Figure 2.6: Format de message DAO-ACK

RPLInstanceID: Un champ de 8 bits indiquant l'instance de topologie associée au DODAG, comme l'a appris le DIO.

D: Indique que le champ DODAGID est présent. Cela serait généralement défini uniquement lorsque un RPLInstanceID local est utilisé.

Reserved: Un champ de 7 bits, réservé aux flags.

DAOSequence: Incrémenté à chaque message DAO à partir d'un nœud, et écho dans le DAO-ACK par le destinataire. Le DAOSequence est utilisée pour corréler un message DAO et un message DAO-ACK.

Status: Indique la fin. Le statut 0 est défini comme une acceptation non qualifiée dans cette spécification. Les valeurs d'état restantes sont réservées en tant que codes de rejet.

Construction de DODAG:

La construction DODAG est basée sur le processus Neighbor Discovery (ND), qui consiste en deux opérations principales:

1. Diffuser des messages de contrôle DIO émis par la racine DODAG pour construire des itinéraires dans la direction descendante de la racine vers le bas vers les nœuds clients,

2. Unicast des messages de contrôle DAO émis par les nœuds clients et envoyés à la racine DODAG pour construire des routes dans la direction ascendante.

Afin de construire un nouveau DODAG, la racine DODAG diffuse un message DIO pour annoncer son DODAGID, son rang pour permettre aux nœuds de déterminer leurs positions dans le DODAG, et la Fonction Objective identifiée par l'OCP (Objective Code Point) dans les champs d'option de configuration au sein du DIO. Ce message sera reçu par un nœud client qui peut être un nœud disposé à se joindre ou un nœud déjà joint. Lorsqu'un nœud désireux de rejoindre le DODAG reçoit le message DIO, il:

- Ajoute l'adresse de l'expéditeur DIO à sa liste parent.
- Calcule son rang en fonction de la fonction Objective spécifiée dans l'OCP déposée, De sorte que le rang du nœud est supérieur à celui de chacun de ses parents.
- Transmettre le message DIO avec les informations de classement mises à jour.

Le nœud client choisit le parent le plus préféré parmi la liste de ses parents comme nœud par défaut par lequel le trafic entrant est renvoyé. Lorsqu'un nœud déjà associé à DODAG reçoit un autre message DIO, il peut procéder de trois façons différentes:

- Rejeter le message DIO selon certains critères spécifiés par RPL.
- Traiter le message DIO pour conserver son emplacement dans un DODAG existant
- Améliorer sa localisation en obtenant un rang inférieur dans le DODAG sur la base du calcul du coût de trajet spécifié par la fonction Objective.

Chaque fois qu'un nœud change de rang, il doit jeter tous les nœuds de la liste des parents dont les rangs sont plus petits que le rang du nouveau nœud calculé pour éviter les boucles de routage.

Si l'indicateur de mode de fonctionnement dans l'objet de base DIO est différent de zéro, les itinéraires descendants de la racine vers les nœuds sont pris en charge et doivent etre conservés. Dans ce cas, chaque nœud client doit envoyer un message de contrôle DAO unicast pour déterminer les informations de route inverse. Lorsque vous retournez à la racine DODAG, les nœuds visités sont enregistrés dans le paquet le long de la route ascendante et un itinéraire complet est ensuite établi entre la racine DODAG et le nœud client.

Exemple de construction d'un DODAG

Supposons que nous avons 5 nœuds comme le montre le figure 2.7: A, B, C, D et E. et qu'ils doivent faire un DODAG, alors les étapes suivantes sont prises:

1. A va diffuser des DIOs. Le reste des nœuds qui reçoivent des DIO essaie de se joindre indépendamment de leurs distances, ils apprennent également que leur distance du nœud A sont 1, 1, 3, 4.
2. B, C, D et E envoient des DAO à A.
3. A les accepte en répondant avec un message DAO-ACK.
4. Maintenant un autre tour commence, et il commence par les nœuds qui sont les plus proches de la racine. Puisque B et C ont un rang de 1, ils commencent à envoyer des

DIO. D reçoit ceux-ci et trouve que sa distance à B et C est respectivement 1 et 2. De façon similaire E reçoit ces DIO, et elle détermine sa distance de B et C pour être égale à 2 et 1 respectivement.

5. Comme D est plus proche de B et E est plus proche de C, D envoie donc DAO à B et E envoie DAO à C.

6. Pour finaliser la formation DODAG, B envoie un DAO-ACK à D, et C envoie un DAO-ACK à E.

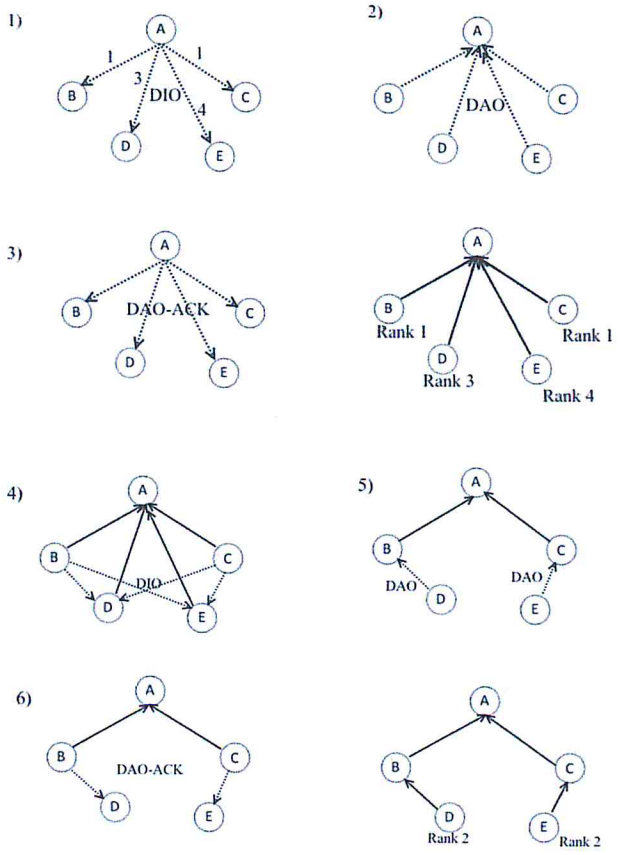


Figure 2.7: Example de formation d'un DODAG

Conclusion

Dans ce chapitre nous avons parlé de l'Internet des objets, les catégories de routage existantes ainsi sur les problèmes tel que les ressources limitées, la topologie de routage dynamique, evolutivité, etc. La fonction d'un protocole de routage est de garantir que les données transmises à partir d'un noeud source du réseau arrivent à un noeud destination.

Nous avons parlés aussi sur le protocole de routage LOAD, et nous avons détaillé le fonctionnement du protocole RPL pour tre l'objet de notre étude.

Dans le chapitre suivant nous allos etudier la QoS dans RPL et quelques fonctions objectives existantes dans le but de faire une extension du protocole RPL pour garantir des exigences de la qualité de service

Chapter 2: Qualité de service dans RPL

Introduction

La performance d'un réseau est un élément fondamental et nécessaire pour une utilisation efficace d'applications, notamment les applications qui exigent une qualité de service (QoS, Quality of Service) telles que le traffic routier, les catastophes, la téléphonie, ou encore les applications temps réels. Le déploiement de telles applications dans les LLNs représente de nombreux intérêts.

Cependant on doit faire face à plusieurs défis et difficultés, et trouver des solutions fiables qui aident à l'intégration de la qualité de service dans ce genre de réseaux. Dans ce chapitre, on s'intéressera aux solutions de la QoS dans les LLNs, plus particulièrement au routage avec QoS, pour ce faire nous présentons d'abord les principales notions de qualitéde service et ces défis, la classification selon plusieurs paramètres et les traveaux connexes a cette denière.

2.1 Que signifie qualité de service (QoS)?

La qualité de service (QoS) désigne la capacité d'un réseau à fournir un meilleur service au trafic réseau sélectionné en termes de bande passante, de jitter et de latence (requis par un trafic en temps réel et interactif) et des caractéristiques de perte améliorées.

2.2 Paramètres de la QoS

Différentes applications ont des exigences différentes concernant la gestion de leur trafic dans le réseau. Les applications génèrent du trafic à des taux variables et exigent généralement que le réseau soit capable de transporter le trafic au rythme auquel il le génère. Certaines applications peuvent tolérer un certain degré de perte de trafic tandis que d'autres ne peuvent pas.

La QoS comporte une série de paramètres qui permettent de caractériser les garanties qui peuvent être fournies à chaque flux (ou connexion) transitant par un réseau de télécommunication. Ces paramètres sont généralement le délai de bout en bout, la bande passante, la gigue, la perte de données.

- **Le délai de bout en bout :** Appelé aussi latence ou temps de réponse, il s'agit de la durée nécessaire à l'acheminement d'un paquet de données de bout en bout. Cette durée dépend de la qualité du support des liaisons, mais aussi du temps passé dans les différentes files d'attente du chemin. Par conséquent, le délai augmente avec la charge du réseau. Par exemple,le trafic routier est sensible au délai.

- **La bande passante :** La bande passante est la quantité maximale de données pouvant etre fransmise d'une source vers une destination en une unité de temps. Il s'agit en fait du minimum des bandes passantes disponibles sur les liens composant le chemin de la source à la destination. La bande passante disponible sur un chemin dépend donc du support des liaisons, mais aussi du nombre et du débit des flux qui partagent ces liaisons.

- **La gigue :** La gigue est la variation du délai de bout en bout, issue des congestions instantanées lorsque plusieurs flux de données sollicitent au même moment le même port de sortie.

- **La perte de données :** Le taux de perte est la proportion des paquets qui ne parviennent pas à leur destination. Ces pertes dépendent de :

  - **La fiabilité du support des liaisons :** un support peu fiable entraîne une

corruption fréquente des paquets. Si un paquet s'avère corrompu à l'issue de sa vérification grâce aux bits de contrôle (checksum), il sera détruit. On peut toutefois estimer qu'actuellement la fiabilité des liens dans le réseau Internet est relativement élevée, et que par conséquent, les pertes de paquets dues à des défauts de support sont très faibles.

- **L'occurrence de surcharges locales (congestions) dans le réseau :** les paquets devant être placés dans une file d'attente pleine sont détruits. Lorsque le protocole de transport garantit l'arrivée de l'information transmise, les paquets perdus doivent être renvoyés. Les pertes ont donc une influence directe sur l'augmentation du délai et sur la gigue.

2.3 Défis de QoS

Les principaux défis dans la QoS sont: [16]

- **La congestion du réseau:** Lorsqu'un réseau est congestionné, le délai de bout en bout augmente car le paquet passe plus de temps dans la file d'attente à chaque saut. La perte augmente également parce que si une file d'attente est trop pleine, elle commence a supprimer les paquets entrants. Ce phénomène limite également le débit puisque les paquets supplémentaires envoyés seront simplement supprimés.

- **Le routage multi-chemin:** Lorsque deux paquets sont envoyés, il n'y a généralement aucune garantie qu'ils prendront le même chemin vers la destination. Si un chemin a plus de saut que l'autre ou est plus congestionné, les paquets n'arrivent pas à la destination en même temps. Ce routage incontrôlé peut provoquer un retard ou une gigue inacceptables.

- **L'évanouissement:** Lorsque plusieurs versions du même signal sont reçus à la destination, ils seront déphasés ou décalés due à l'effet doppler, ils peuvent interférer les uns avec les autres.

- **Le retard de propagation:** Certains réseaux sans fil couvrent des distances mesurées en kilomètres. Dans ces réseaux, le retard de propagation peut constituer un énorme fardeau pour toute communication, mais surtout pour la communication qui exige une garantie de délai total.

Enfin, il peut être difficile de maintenir des garanties de service dans un réseau.

2.4 Motivation pour RPL

RPL peut fonctionner dans plusieurs conditions, il est flexible en ce qui concerne les règles pour former des topologies

RPL a été spécialement conçu pour répondre aux besoins des nœuds à ressources limitées. Il supporte principalement le trafic multipoint à point (MP2P), avec un support raisonnable pour le trafic point à multipoint (P2MP) et les fonctionnalités de base pour le trafic point à point (P2P).

En raison de telles considérations de conception, RPL a pu rester un conservateur en termes de ressources limitées.

Les protocoles de routage existants tels que OSPF [8], IS-IS [12], AODV et OLSR [67] ont fait l'objet d'une évaluation approfondie par le groupe de travail ROLL (ROLL-WG) [62] et finalement ils n'ont pas satisfaient les exigences des LLNs [41]. Par exemple, la sélection de chemin doit être conçue pour prendre en compte les capacités de puissance, les attributs et les caractéristiques fonctionnelles spécifiques des liaisons et nœuds du réseau.

2.5 Classification

Plusieurs métriques de routage ont été recommandés pour les LLN. Cependant, la spécification de RPL n'a pas imposé de métrique de routage et la laissé ouvert aux implémentations. Les fonctions objectives proposées par l'IETF (OF0 et MRHOF) ont défini certaines recommandations sur la façon de Leurs implémentations sans préciser les paramètres de routage qui

peuvent être utilisés.

Nous proposons une classification pour les différentes propositions et implémentations, voir la figure 3.1

- **Routage**

  unipath : Les protocoles de chemin unique apprennent les itinéraires et choisissent un meilleur chemin pour chaque destination. Ces protocoles sont incapables d'équilibrer la charge entre les nœuds.

  multipath : Les protocoles multi-chemins apprennent des itinéraires et peuvent sélectionner plus d'un chemin vers une destination. Ces protocoles sont préférables pour effectuer un équilibrage de charge du trafic entre les nœuds.

- **Métrique**

  certaines fonctions objectives peuvent utiliser plus d'une métrique dans le processus de sélection parental, tandis que d'autres peuvent utiliser une seule métrique (ETX, nombre de sauts ...).

- **Mobilité**

  RPL est conçu pour les réseaux statiques. Cependant, de nombreuses applications LLN nécessitent un support de routage pour les nœuds mobiles. La mobilité augmente considérablement la dynamique des réseaux. Par exemple, le parent préféré d'un nœud est temporairement indisponible, ce qui l'oblige à supprimer les paquets et à modifier le parent préféré. Ce processus augmente considérablement la latence de la transmission de données. Des modifications nécessaires sont requises pour que RPL prenne en charge des réseaux composés de nœuds mobiles totalement ou partiellement

- **Communication**

  Unicast : est un type de communication où il n'y a qu'un seul expéditeur et un seul récepteur.

  Multicast : est un type de communication où le même paquet de données est envoyésimultanément à plusieurs récepteurs dans le netowrk

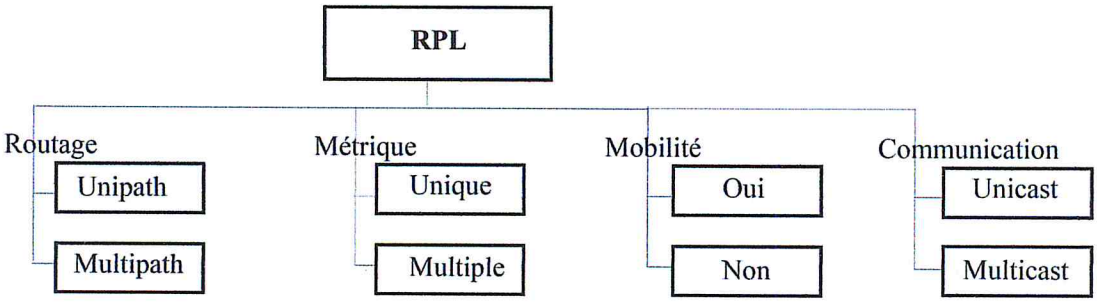


Figure 3.1: Diagramme de classification

2.6 Traveaux connexes

De nombreuses mesures de routage et des fonctions objectives ont t proposes pour tre utilises dans le protocole de routage RPL dans les rseaux LLN et 6LoWPAN. Dans cette section, nous Discutons ces paramtres de routage et ces fonctions objectives.

1. **OF0 :** La fonction Objective par défaut dans RPL, OF0 est simple, elle sélectionne son parent en fonction des rangs minimaux des voisins.Cette fonction objective permet au trafic ascendant d'être acheminé par le parent sélectionné (parent préféré) sans effectuer d'équilibrage de charge.[22]

2. **MRHOF :** La fonction objective d'hystérésis (MRHOF) utilise plusieurs métriques qui sont présentes dans le conteneur métrique DAG, pour déterminer le "meilleur" chemin. [8]

3. **LQL :** LQL (Link Quality Level) est utilisé pour quantifier la fiabilité du lien à l'aide d'une valeur discrète, de 0 à 7, où 0 indique que le niveau de qualité de la liaison est inconnu et 1 indique le niveau de qualité de liaison le plus élevé.[23]

4. **ELT :** La métrique de durée de vie prévue (ELT) c'est combien de temps un capteur doit-il vivre avant que son batterie soit déchargée et montre comment l'estimer pour chaque nœud.[11]

5. **OF-FL :** Une nouvelle fonction objective pour RPL basée sur la Logique Floue qui surmonte les limitations de la norme RPL et permet de sélectionner les meilleurs chemins vers la destination. [5]

6. **SCAOF :** Donne une règle fondamentale qui définit comment utiliser des informations contextuelles pour calculer le classement et la sélection des parents préférées. Ainsi, SCAOF peut gérer la construction initiale et la mise à jour occasionnelle d'un arbre RPAL DODAG. Il est particulièrement conçu par des méthodes / contraintes de routage composite et c'est une approche hautement évolutive et flexible. Par conséquent, SCAOF est en mesure d'utiliser Mrhof standardisé de manière plus contextuelle.[3]

7. **CAOF :** Capture le contexte des nœuds des capteurs définis par les attributs comme: Batterie drainée, connectivité non continue, colocation (L'emplacement d'un nœud voisin par rapport au nœud raçine) pour rendre le RPL plus souple pour les pannes de nœuds et de fonctionner plus efficacement avec les changements de contexte dans les réseaux de capteurs fonctionnant judicieusement avec leurs ressources limitées dans les décisions de routage.[19]

8. **ALABAMO :** Cette OF utilise à la fois le profil de trafic des nœuds et l'ETX des liaisons pour résoudre le problème de déséquilibre. La proposition est appelée ALABAMO (A LoAd BAlancing MOdel pour RPL) et est un modèle flexible qui utilise deux paramètres d'entrée et améliore la tche de choix le parent préféré dans RPL vìsant a l'extension de la durée de vie du réseau.[14]

9. **Cyber-OF :** La fonction Cyber-Physical objective (Cyber-OF) est conçu pour adapter la structure arborescente du réseau en temps réel aux propriétés cyber-physiques de l'environnement en fonction de la criticité de l'événement. En effet, pour les paquets de données normaux, l'objectif est de maximiser la durée de vie du réseau, ainsi, la fonction objective optimise la consommation d'énergie. En cas d'événement critique, le réseau doit adapter sa topologie pour minimiser les retards de bout en bout.[6]

Table 3.1: Tableau comparatif

proposition	métriques	date	mplémentation	applications	avantages	inconvénients
OF0	rank	2012	* fonction objective par défaut	Nombreuses applications	simple	* ne favorise pas des liens de haute qualité.*pas d'équilibrage de charge. * ne prend pas la qualité de service*peut générer le chemin erroné pour un paquet, ce qui a pour résultat d'être mal acheminé.
MRHOF (The Minimum Rank with	*ETX*smallest path cost* MRHOF may be used with any additive metric		TinyRPL	Nombreuses applications	"*Minimise la consommation d'énergie par nœud. * Minimise les cots de chemin,"	"* la distribution d'énergie restante n'est pas équilibrée *ne tienne pas compte des exigences des applications dans le processus d'optimisation de chemin."
LQL (Link Quality Level) niveau de qualité de liaison	le niveau de qualité de lien LLQ ETX			amélioration en termes de taux de perte de paquets et de latence moyenne du réseau.	*consommation d'énergie élevée et un taux de perte de paquets élevé dans le cas de taux de transmission élevés*cause une détérioration de la performance	
ELT (rexpected lifetime)	*reliability *delay *energy consumption	2014	*simulated using WSNet		*réduire le nombre de changements parents*Minimise la consommation d'énergie	n'offre pas une bonne performance en termes de délai moyen de bout en bout et de taux de livraison de paquets
OF-FL (Objective Function based on Fuzzy Logic)	*ETX (Expected transmission count)*hop count(Nombre de sauts*end-to-end-delay(bout en bout) * niveau de la batterie* LQL	2014			*réagir aux changements fréquents de la topologie.*permet de sélectionner le candidat le plus efficace.*une grande amélioration dans le réseau basé sur RPL.	*Réseau instable à cause de changement fréquant des parents.*N'est pas adaptée pour les WSN hétérogènes

SCAOF (Scalable Context-Aware Objective Function)	*Hop count (HC) :Nombre de sauts, *la transmission attendue (ETX), *l'indicateur de transfert de paquets (PFI),l'énergie restante (RE)	2015	*Agricultural Low-Power andLossy Networks (RPAL)	surveillance environnementale agriculture	*améliore le taux de livraison et garantit plus d'équité en termes d'exploitation de la batterie pour les différents nœuds.*atténuer le problème des hotspots et améliorer la QoS de A-LLN en combinant l'énergie restante des nœuds de capteurs A-LLN, en tenant compte des autres facteurs.*Scalable	*peut provoquer des flux extra dans le réseau en raison de la sélection d'un nouveau chemin de routage *Produit quelque occillations dans des larges DODAG
CAOF(Context-Aware Objective Function)	*hop-count* les transmissions attendues en termes de débit réseau* durée de vie du réseau	2014		*la surveillance environnement / agriculture *automatisation des btiments	*prend en compte des ressources limitées des nœuds capteurs et de leurs modifications temporelles. *améliore le taux de livraison et garantit plus l'équilibrage d'exploitation de la batterie pour les différents nœuds. *rendre le RPL plus souple aux échecs de nœuds et fonctionne plus efficacement avec les changements de contexte.	
FMOF(Fuzzy Mobility Objective Function)la fonction objectif de mobilité floue	ETX(Expected transmission count)RSSI (Received signal strength indicator)Hop-count	2016	*Low-Power Wireless Networks (LPWNs)	*surveillance de la santé *l'automatisation industrielle et les villes intelligentes	*Les itinéraires sont mis à jour plus fréquemment sur la base d'informations plus précises générées par le FMOF optimisé. , *améliore le délai moyen de transmission,*améliore le taux de réception de paquets.*fournir une bonne connectivité en terme de délai.	Pas implémenté dans un environnement réaliste
ALABAMO(A LoAdBAlancing MOdel for RPL)	*the traffic profile of the nodes(le profil de trafic des nœuds)*ETX	2016		*application de collecte de données multi-hop, comme la surveillance environnementale	*compatible avec RPL et permettant un acheminement équilibré et respectueux de la circulation**Ameliorer la durée de vie du réseau.	Il réduit un peu le taux de livraison du réseau.
Cyber-OF (Cyber-Physical objective function)	*End-To-End delay(délai de bout en bout) *Energy Metric	2016	ContikiOS /Cooja simulator	*Smart City *catastrophes météorologiques *Accidents	*comportement adaptatif: améliorer la durée de vie du réseau dans le cas normal et minimiser les retards de bout en bout en cas d'événement critique.	

Conclusion

Dans ce chapitre nous avons parlé sur la notion de la QoS, et ses défis et les diffrentes techniques permettant de garantir une QoS dans les réseaux. Ces techniques interviennent diffrents niveaux.

Nous avons expliqué la motivation pour protocole RPL, et nous avons donné une description sur quelques fonctions objectives existantes avec un tableau comparatif qui englobe toutes les informations necessaires.

Dans le chapitre suivant nous allons étudier les détailles de la conception d'un protocole de routage avec QoS ; il s'agit d'une extension du protocole RPL étudié en détails dans le chapitre 2 on lui rend sensible á une métrique de QoS.

Chapitre 3:Conception de la métrique D-RPL

Introduction Dans le cadre de cette étude nous avons choisi l'Approche évolutionnaire qui consiste à faire des extensions des protocoles existants ou d'apporter des améliorations à un protocole de routage avec QoS en ajoutant d'autres métriques par exemple.

On à choisi cette approches pour deux raisons ; la première qu'il est plus efficace et facile et encore moins coûteux d'améliorer un travail existant. La deuxième est qu'une extension d'un protocole existant est plus compatible avec la version originale ce qui permet l'utilisation des deux à la fois et facilite l'interconnexion des deux plate formes. Le protocole concerné par cette étude est le protocole RPL étudié en détail dans le chapitre II.

Les chemins établis par le protocole RPL standard ne permettent pas de garantir certains critères de qualité de service. C'est pourquoi il semble important de faire une extension de ce protocole afin d'assurer une certaine qualité de service. la QoS dans RPL peut être integré dans RPL par l'introduction de la métrique "délai" qui est plus appropriée que la distance (nombre de sauts).

Pour ce faire, nous avons commencé d'abord par un diagramme de séquence expliquant le fonctionnement global de RPL, ensuite par de schémas montrons le fonctionnement détailléau cœur de RPL, aprés nous expliquons les méthodes de calcul et les modifications nécessaires ajoutées à la structure de RPL ainsi que le mécanisme de fonctionnement basé sur la recherche de routes assurant la métrique délai de bout en bout.

جامعة البليدة ١ - مكتبة الكلية - كلية العلوم

3.1 Diagramme de séquence pour le protocole RPL

Tout d'abord, lorsque le nœud se réveille, il initialise RPL, réinitialise les temporisateurs appropriés et envoie un message DIS sollicitant un message DIO du voisin le plus proche. Un autre nœud recevra ce message DIS et répond avec un message DIO. Le premier nœud reçoit le message DIO et traite ce message pour rejoindre le DAG de l'expéditeur DIO et choisir le meilleur parent des parents candidats.
Une fois que le nœud décide qu'il rejoindra ce DAG, il choisira un parent préféré, selon l'OF proposé, et connaîtra son chemin par défaut vers la racine DAG. Ensuite, il enverra un message DAO vers le haut pour s'annoncer sur le réseau et il sera reconnu par DAO -ACK
La figure 4.1 montre le schéma de séquence du scénario ci-dessus.

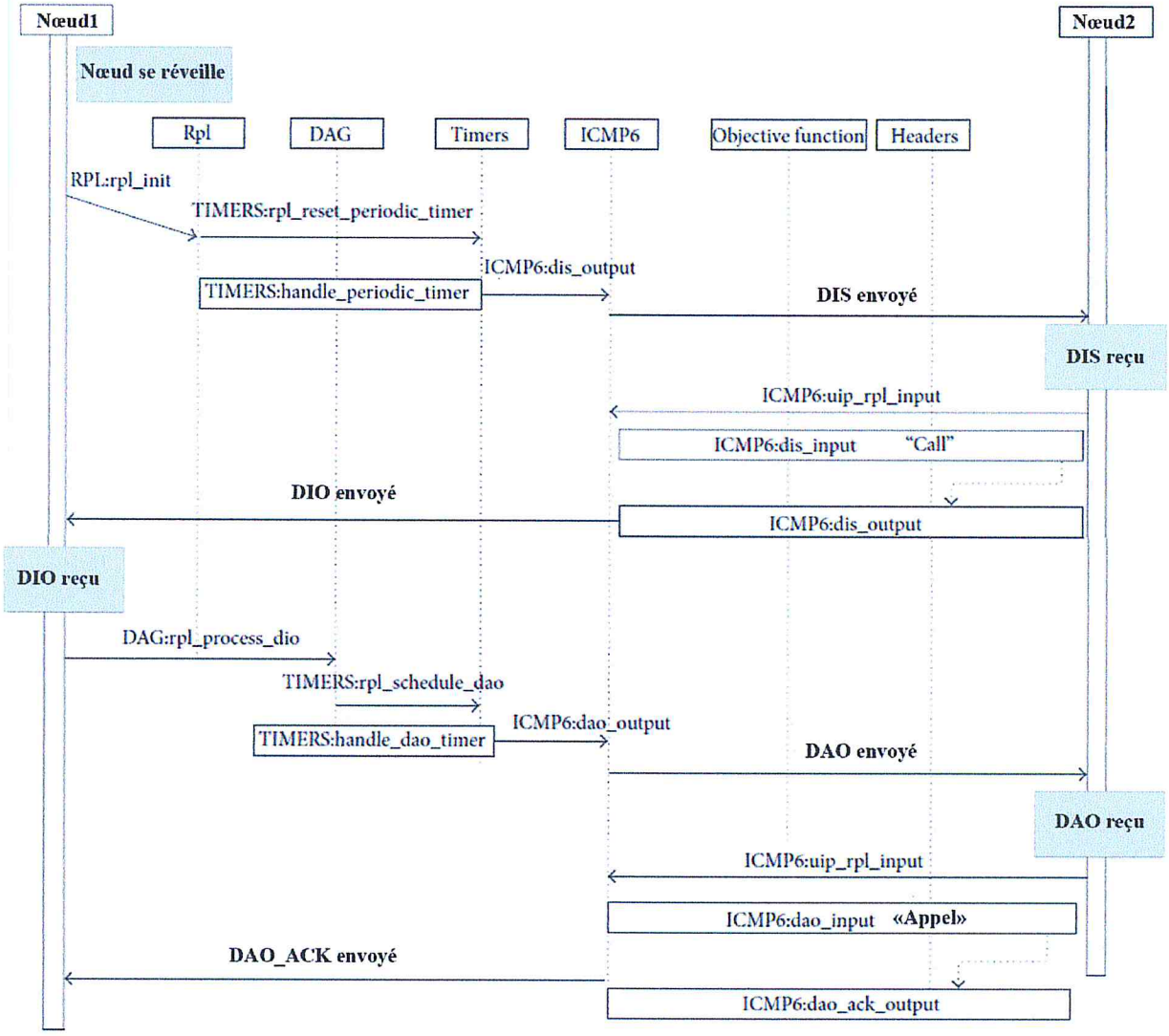


Figure 4.1: Diagramme de séquence

Une fois la transmission des données est en progression, les chemins sont renforcés positivement, ce qui rend plus souhaitable une sélection supplémentaire. De même, lorsque la session se poursuit, la charge sur le chemin sélectionné peut augmenter, ce qui provoque plus de délai et moins de bande passante disponible.

Les nœuds pourraient échouer (épuisement de leur énergie) provoquant la rupture du lien. Dans ce cas, la probabilité de préférence de chemin diminue automatiquement et, par conséquent,

des routes alternatifs qui se trouvent au cours de la phase de construction DAG peuvent être utilisés . Les routes alternatifs sont également périodiquement vérifiés pour leur validité même s'ils ne sont pas actuellement utilisés.

3.2 Construction et maintenance des DODAGs

Le routage avec QoS est un processus d'etablissement et de maintenance des routes satisfaisant un ensemble de critères quant à la qualité de la transmission des données. La construction et la maintenance des DODAGs sont réalisées grâce à des messages de contrôle ICMPv6 (DIS, DIO, DAO).
Un nouveau nœud peut rejoindre un réseau déjà formé en diffusant un message DIS pour solliciter en réponse un message DIO qui contient des informations sur le DODAG comme le numéro de version et l'identifiant du DODAG, l'identifiant de l'instance et l'OF utilisée.

Un nœud peut également attendre de recevoir un message DIO diffusé périodiquement par ses voisins. La fréquence d'envoi des messages DIO est déterminée par un temporisateur fondé sur l'algorithme Trickle (appelé également temporisateur Trickle). à la moindre anomalie dans le réseau, le temporisateur Trickle est réinitialisé pour permettre à la topologie de reconverger.

La construction DODAG est basée sur le processus Neighbor Discovery (ND), qui consiste en deux opérations principales :

- ´emission de messages de contrôle DIO émis par la racine DODAG pour générer des itinéraires vers le bas de la racine vers les nœuds clients,
- Unicast de messages de contrôle DAO émis par les nœuds clients et envoyé à la racine DODAG pour créer des itinéraires vers le haut.

Pour construire un nouveau DODAG, la racine DODAG diffuse un message DIO pour annoncer son DODAGID, son rang d'information permettant aux nœuds de déterminer leurs positions dans le DODAG et la fonction objective identifiée par le Objective Code Point

(OCP) dans le DIO Champs d'option de configuration.
Ce message sera reçu par un nœud client qui peut être soit un nœud disposé à se joindre soit un nœud déj à joint. Lorsqu'un nœud disposé à rejoindre le DODAG reçoit le message DIO, il :

- ajoute l'adresse de l'expéditeur DIO à sa liste parentale,
- calcule son rang selon la fonction objective spécifiée dans l'OCP déposée, de sorte que le rang du nœud est supérieur à celui de chacun de ses parents, et
- transmettre le message DIO avec les informations de classement mises à jour. Le nœud client choisit le parent le plus préféré parmi la liste de ses parents comme nœud par défaut par lequel le trafic vers l'intérieur est renvoyé.

Lorsqu'un nœud déjà associé à DODAG reçoit un autre message DIO, il peut procéder de trois façons différentes :

- rejeter le message DIO selon certains critères spécifiés par RPL,
- traiter le message DIO pour maintenir son emplacement dans un DODAG existant Ou
- améliorer sa localisation en obtenant un rang inférieur dans le DODAG en fonction du calcul du coût du trajet spécifié par la fonction objective. Chaque fois qu'un nœud change de rang, il doit jeter tous les nœuds dans la liste des parents dont les rangs sont plus petits que le rang du nouveau nœud calculé pour éviter les boucles de routage.

L'organigramme présenté dans la Figure 4.2 résume l'opération d'un routeur dans un DODAG.

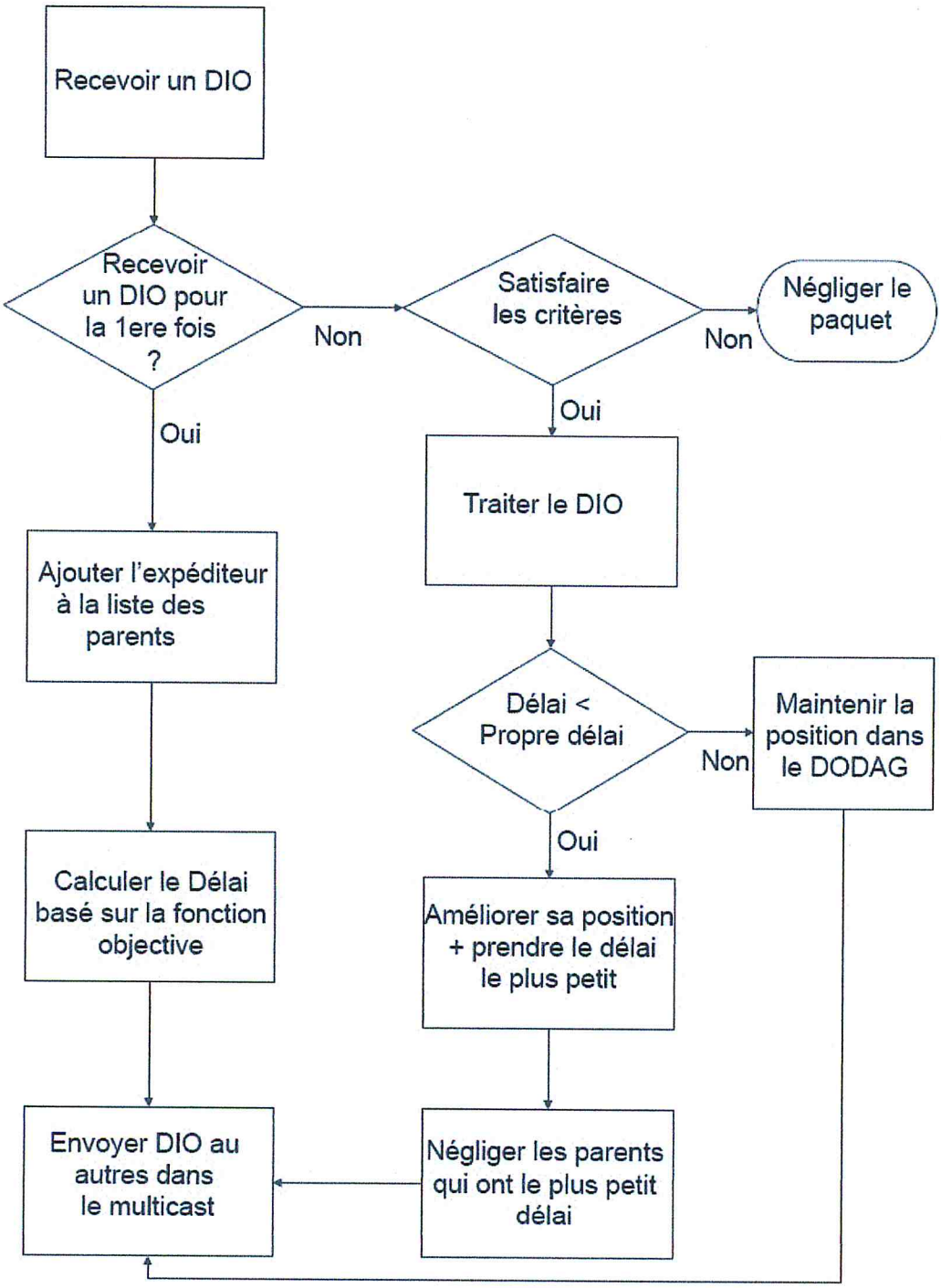


Figure 4.2: Opération d'un routeur dans un DODAG

3.3 La latence

La latence provient généralement de deux sources différentes: les routeurs et la distance. Chaque routeur sur lequel se déplace un paquet doit copier le paquet d'une interface réseau vers l'autre. Ceci introduit un très léger retard: typiquement quelques millisecondes. Toutefois, le trafic sur le réseau pourrait avoir à parcourir plus de cinquante routeurs lors d'un voyage aller-retour entre deux nœuds, de sorte que les retards s'ajoutent. Les routeurs et les réseaux occupés qui sont près de la saturation peuvent introduire plus de latence car le routeur doit attendre plusieurs millisecondes avant de pouvoir placer un paquet sur une interface réseau. L'outil le plus courant pour mesurer la latence est le Ping, où un certain nombre de messages hello sont envoyés à une racine et la latence des paquets est calculée à la racine. Ping peut vous donner une idée approximative de la latence entre deux points, mais il consomme plus d'énergie et génère plus de trafic qui peut augmenter la latence du réseau. C'est pour cette raison on à préférés d'utiliser le message de controle DIO pour le calcul áu lieu d'utliser le ping dans notre etude pour éviter l'augmentation du traffic.

3.4 Estimation du délai à un saut

Le délai de bout en bout est une métrique additive. Il est égal à l'ensemble des délais à un saut sur le chemin menant de la source à la destination. Ce dernier peut se décomposer en deux parties : Le délai entre l'instant où le paquet entre dans la file d'attente du nœud émetteur et l'instant où il est passé à la couche MAC. Le délai s'écoulant entre le moment où le paquet est reçu par la couche MAC jusqu'à la réception de l'acquittement correspondant par le nœud récepteur, Un paquet se trouvant à une station quelconque provient de deux sources : les paquets générés localement au niveau du nœud considéré et les paquets routés qui passent par ce nœud. Ainsi, chaque nœud du réseau peut jouer le rôle de nœud source, routeur ou destination

3.5 Délai multi-sauts

Le délai est une métrique additive, autrement dit le délai moyen de bout en bout entre une source s et une destination d noté $D_{s,d}$, est égal à la somme des délais D_i moyens des liens constituants ce chemin.

$$D_{s,d} = \sum_{i \in [s,d]} D_i \tag{4.1}$$

3.6 Intégration dans PRL

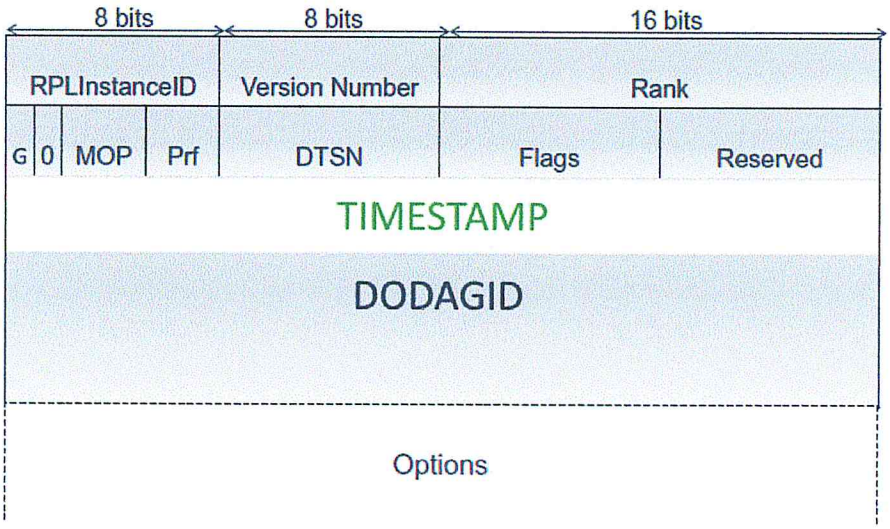


Figure 4.3: extension de message DIO

nous avons modifiés le format de message DIO, en ajoutant un nouveau champs qui est le "timestamp" comme le montre la figure4.3, ce champs nous permet de connaitre le temps d'envoi d'un message; et pour savoir le temps de la réception d'un message on à utiliser le "clock time" qui renvoi le temps courant d'un message, le délai à un saut est obtenu en soustrayant le temps d'envoi qui est le "timestamp" à partir du temps de réception de message qui est le temps courant.Comme le montre l'équation 4.2

$$delai = temps \quad courant - timestamp \tag{4.2}$$

3.7 Limitations

La partie touchée par les modifications concerne seulement la phase de découverte de routes. En effet, la solution présentée assure un certain degré de qualité de service de telle sorte que toutes les routes établies respectent la métrique délai de bout en bout. Cependant aucune garantie n'est fournie pendant la transmission des paquets de données. Si le délai augmente sur un chemin après la validation de ce lui ci, aucun mécanisme de détection et de maintenance de telles situations n'est fournie.

Conclusion

Ce présent chapitre décrits les spécifications d'une solution qui permet d'étendre le protocole RPL pour garantir de la QoS en termes de délai de bout en bout. Il détaille également le principe de fonctionnement du nouveau protocole et la méthode d'estimation du délai utilisée. Ce chapitre a été dédié à la conception de RPL. nous y avons abordé les principaux schémas qui montrent les relations entre les différentes fonctions du protocole.

Le prochain chapitre sera consacré aux détails de l'implémentation du protocole D-RPL modifié sous le simulateur Cooja.

Chapitre 4 : Implémentation de la métrique D-RPL

Introduction

Ce présent chapitre est consacré aux détails de l'implémentation du protocole D-RPL sous le simulateur Cooja et le système d'exploitation Contiki. Pour ce faire, nous présentons d'abord les différents fichiers et modules concernés par les modifications. Ensuite nous décrivons les modifications nécessaires du protocole RPL pour intégrer la méthode d'estimation du délai adoptée, et enfin, les nouvelles méthodes ajoutées, les changements dans le format des paquets, et le mécanisme de fonctionnement dans D-RPL.

4.1 Apercu sur le système d'exploitation

Contiki est un système d'exploitation de réseau sans fil et se compose du cœur, les bibliothèques, le chargeur de programme et un ensemble de processus. Il est utilisé dans les systèmes embarqués interconnectés et les objets intélligents.

Contiki fournit des mécanismes qui aident à la programmation des applications d'objet intélligents. Il fournit des bibliothèques pour l'allocation de la mémoire, la manipulation de listes chaînées et les abstractions de communication. C'est le premier système d'exploitation qui a fourni la communication d'IP. Il est développé en C, toutes ses applications sont aussi développées dans le langage de programmation C et donc c'est extrêmement portatif à de différentes architectures comme Texas Instruments MSP430.

Contiki est un système déclenché par événement dans lequel les processus sont exécutés

comme les dresseurs d'événement qui s'éxécutent à l'achèvement. Un système Contiki est divisé en deux parties : le cœur et les programmes chargés. Le cœur se compose du cœur Contiki, le chargeur de programme, la durée d'exécution, et une pile de communication avec les pilotes de périphériques pour le matériel de communication .

Le chargeur de Programme charge les programmes dans la mémoire et il peut l'obtenir d'un hôte en utilisant la pile de communication ou il peut l'obtenir depuis le périphérique de stockage attaché comme EEPROM.

Le système d'exploitation Contiki fournit des modules à de différentes tâches (les couches). Il fournit les modules acheminants dans un répertoire séparé contiki/core/net/rpl et se compose d'un certain nombre de fichiers. Ces fichiers sont séparés logiquement basés sur les fonctionnalités qu'ils fournissent, par exemple rpl-dag.c contient la fonctionnalité pour la formation de DAG, rpl-icmp6.c fournit la fonctionnalité emballage pour les messages d'ICMP etc.

4.2 Aperçu sur le simulateur

Pour effectuer nos tests d'évaluation, nous avons utilisé le simulateur Cooja[18]. Cooja (Contiki os java simulator) a été développé en 2002 à l'institut de recherche suédois SICS ICT [119]. Cooja est reconnu pour être un simulateur bien développé, destiné à la simulation des réseaux de capteurs connectés à l'IoT (les réseaux 6LoWPANs). Il est open source et portable . Cooja appartient à la famille des simulateurs àevènements discrets, son code est écrit en langages C et Java et repose sur le concept de protothreading [128].

Etant destiné à la simulation des réseaux 6LoWPANs, Cooja implémente une pile protocolaire TCP/IP complète et adaptée pour les environnements capteurs (IPv6, 6LoWPAN, RPL, ICMPv6, TCP, UDP, etc.). Ainsi, Cooja définit plusieurs émulateurs de capteurs, à savoir cooja mote, TMote Sky, Z1, Micaz, etc. avec un modèle de simulation détaillé et très proche de la réalité des capteurs. Cooja supporte même la simulation des réseaux de capteurs

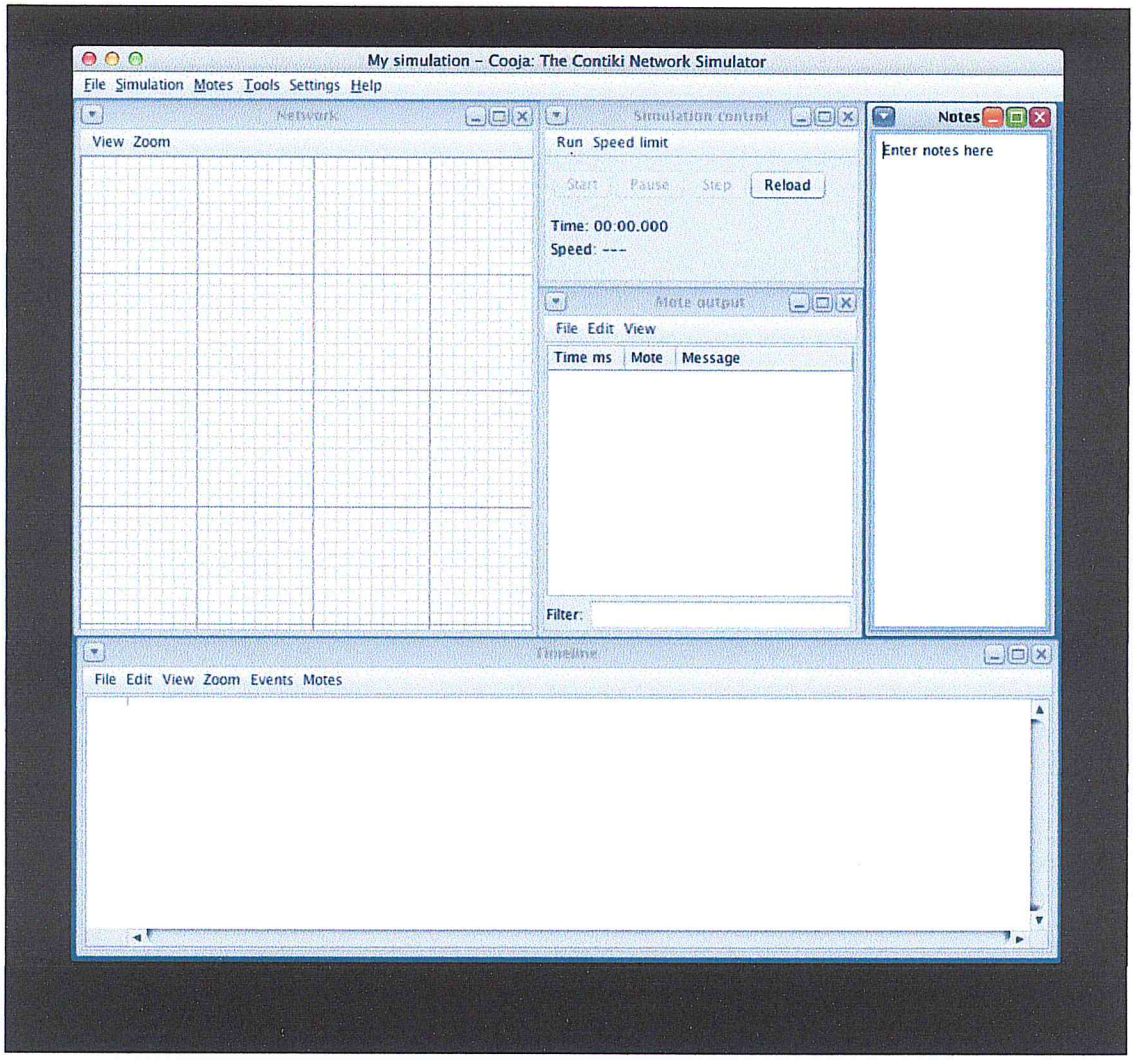


Figure 5.1: L'interface de simulation

hétérogènes regroupant différents types de plateformes capteurs ce qui présente un avantage majeur.

L'interface de simulation, montrée dans la Figure 5.1 , se compose de cinq Fenêtres. La fenêtre Network montre la disposition physique du réseau, c'est-à-dire vous serez capables de placer physiquement des nœuds et les déplacer, selon le besoin, pour former la topologie et la disposition à laquelle vous vous intéressez.

La fenêtre Simulation Control vous permet de commencer, arrêter et recharger la simulation. Il vous permet aussi de contrôler le taux auquel la simulation procède.

La fenêtre Mote Output montre n'importe quelle production série produite par tous les nœuds, c'est-à-dire la production de la commande de printf. Vous pouvez filtrer la production montrée basé sur une chaîne que vous entrez dans le champ de "Filter". Par exemple, si vous voulez filtrer la production tel qu'il montre seulement la production de nœud 2, alors vous pouvez entrer "ID:2" dans ce champ.

La fenêtre Timeline, sert à la visualisation des communications radio (transmission, réception, collision) ainsi que les états réveillé et endormis des nœuds capteurs.
La fenêtre Note est l'endroit où nous pouvons mettre des notes pour notre simulation.
Cooja inclut d'autres fenêtres pas moins importants et peut même être étendu pour en avoir plus.

4.3 Détails de l'implémentation de D-RPL

Contiki OS implémente une version assez stable de RPL Le code source du routage RPL se trouve dans / core / net / rpl c'est dans ce module qu'on va consacrer notre etude car c'est là où on va implémenter notre travail.
Le répertoire RPL contient plusieurs fichiers implémentant différentes fonctions RPL.

Rpl-conf.h : Déclaration de configuration publique et d'API pour ContikiRPL.

Rpl.h : Déclarations d'API publiques pour ContikiRPL.

Rpl-private.h : Déclarations privées pour ContikiRPL.

Rpl-timers.c : Est responsable de l'envoi de mises à jour périodiques.

Rpl-dag.c : Contient les fonctions nécessaires pour les manipulations du DAG.

Rpl.c : Les fonctions RPL principales pour modifier les entrées de routage IPV6 (élimination de route, ajout de route, etc.)

Rpl-icmp6.c : Implémente les fonctions de message ICMP pour le protocole RPL,implémente également des fonctions d'entrée pour traiter les messages de contrôle entrants.

Rpl-of0.c : Fonction Objective 0 de RPL. (la fonction objective de routage basée sur le saut).

Rpl-mrhof.c : Contient la fonction objective mrhof.

Rpl-ext.c : Gestion des en-têtes d'extension pour ContikiRPL.

4.4 Fichiers et modules concernés par les modifications

Afin d'utiliser la métrique de latence, 5 fichiers ont été modifiés: rpl-conf.h, rpl-private.h, rpl.h, rpl-icmp6.c, rpl-mrhof.c

Fichier rpl-conf.h: dans ce fichier nous avons changé la configuration de la métrique ETX par la nouvelle métrique qui est le délai.

```
55  55   #ifdef RPL_CONF_DAG_MC
56  56   #define RPL_DAG_MC RPL_CONF_DAG_MC
57  57   #else
58      -#define RPL_DAG_MC RPL_DAG_MC_ETX
    58  +#define RPL_DAG_MC RPL_DAG_MC_LATENCY //pour utiliser la latence comme métrique
59  59   #endif /* RPL_CONF_DAG_MC */
```

Figure 5.2: Chagment de métrique utilisé

Fichier rpl-private.h: Dans ce fichier nous avons ajouté le champs "timestamp" dans l'objet de message DIO.

```
218  218   /* Logical representation of a DAG Information Object (DIO.) */
219  219   struct rpl_dio {
     220 +   uint32_t timestamp; //champs qui contient le temps d'envoi  de dio
220  221     uip_ipaddr_t dag_id;
221  222     rpl_ocp_t ocp;
222  223     rpl_rank_t rank;
```

Figure 5.3: Ajout du champ de latence dans dio

Fichier rpl.h: chaque nœud diffuse sa latence de chemin au autres nœuds, cette latence se trouve dans la métrique container pour cela nous avons ajouté la métrique latence dans le conteneur de métriques pour qu'il soit diffusé et utilisé au lieu etx.

```
100  100    union metric_object {
101  101      struct rpl_metric_object_energy energy;
102  102      uint16_t etx;
     103 +    uint16_t latency; //pour ajouter la latence dans le conteneur de métriques
103  104    } obj;
104  105  };
```

Figure 5.4: Conteneur de métriques

Fichier rpl-icmp6.c : Dans ce fichier, il existe les deux fonctions responsables de l'envoi et de la réception de messages dio. Dans la fonction dio_output on à mis le temps d'envoi dans le champs timestamp:

```
     464 +
     465 +  uint32_t time_sent=clock_time(); //temps d'envoi
     466 +  set32(buffer, pos, time_sent); //mettre le temps dans le buffer de dio
     467 +  pos += 4;
     468 +
459  469    buffer[pos++] = instance->dtsn_out;
```

Figure 5.5: Timestamp dans dio_buffer

puis on doit récupérer le temps d'envoi depuis le message DIO au niveau de la fonction dio_input:

```
276  276      dio.preference = buffer[i++] & RPL_DIO_PREFERENCE_MASK;
277       -
          277 +  dio.timestamp = get32(buffer, i); //récupérer le temps d'envoi  depuis le message dio
          278 +  i += 4;
278  279      dio.dtsn = buffer[i++];
```

Figure 5.6: Récupérer le temps d'envoi

Après avoir reçu le dio, nous devons calculer la latence en soustrayant le temps d'envoi(timestamp) à partir du temps courant.

```
...  ...  @@ -331,6 +331,29 @@ dio_input(void)
331  331                  (unsigned)dio.mc.prec,
332  332                  (unsigned)dio.mc.length,
333  333                  (unsigned)dio.mc.obj.etx);
          334 +             } else if(dio.mc.type == RPL_DAG_MC_LATENCY) {
          335 +       uint32_t time_now=clock_time();
          336 +       uint16_t latency;
          337 +       latency= time_now - dio.timestamp;
          338 +       instance_id = dio.instance_id;
          339 +       instance = rpl_get_instance(instance_id);
          340 +if(instance == NULL) {
          341 +    PRINTF("instance null (%u)\n",
          342 +           instance_id);
          343 +  }else {
          344 +       dag = instance->current_dag;
          345 +        if(dag == NULL) {
          346 +          PRINTF("dag NULL\n");
          347 +
          348 +  }
          349 +       p = rpl_find_parent(dag, &from);
          350 +       if(p == NULL) {
          351 +                  PRINTF("parent NULL\n");
          352 +        }else{
          353 +        update_latency(p, latency);//latence de noeud au parent
          354 +       }
          355 +}
          356 +        dio.mc.obj.latency= get16(buffer, i + 6); /**latence envoyé par le parent**/
334  357          } else if(dio.mc.type == RPL_DAG_MC_ENERGY) {
```

Figure 5.7: Calculer la latence

Dans ce fichier nous avons ajouté la fonction update_latency qui permet de mettre à jour la latence. Cette fonction va être appellé par la fonction dio_input qui se trouve dans icmp6.c a chaque fois qu'on reçoit un message DIO.

Fichier mrhof.c : MRHOF fonctionne avec des métriques qui sont additifs le long d'une route et la métrique utilisée est déterminée par les métriques annoncé par les messages DIO, et puisque la latence est une métrique additive, on peut l'utiliser avec MRHOF.
À chaque fois qu'on reçoit un nouveau message DIO on appelle la fonction update_latency qui se trouve dans le fichier mrhof.c pour mettre à jour la latence.

```
+void
+update_latency(rpl_parent_t *p, int latency)
+{
+uint16_t recorded_latency = p->link_metric;//latence précédents
+uint16_t packet_latency = latency;//nouvelle latence
+uint16_t new_latency;
+
+new_latency = ((uint32_t)recorded_latency * LATENCY_ALPHA +
+               (uint32_t)packet_latency * (LATENCY_SCALE - LATENCY_ALPHA)) / LATENCY_SCALE;
+p->link_metric = new_latency;
+}
+
+
```

Figure 5.8: Calculer la latence

La latence vers la racine est calculée dans la fonction calculate_path_metric en faisant la somme de métrique de lien sélectionnée(latence calculée) et le coût du chemin annoncé par le parent(latence difusée dans le contenneur métrique).

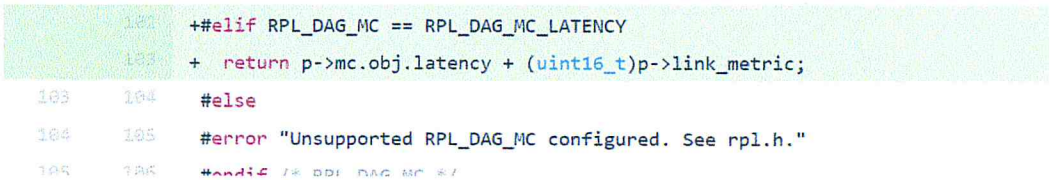


Figure 5.9: La somme vers la racine

Après, le nœud annoncera son coût de chemin qui a était calculé dans la fonction calculate_path_metric en l'envoyant dans le conteneur métrique:

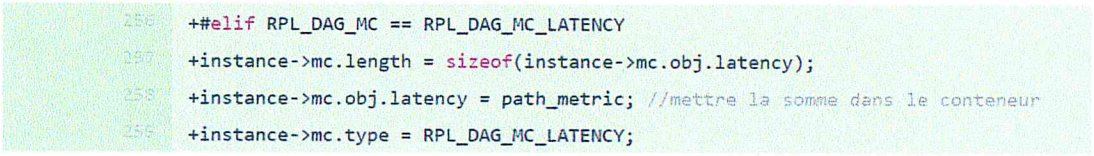


Figure 5.10: Mise à jour du conteneur de métriques

Conclusion

Ce chapitre a été consacré aux détails de l'implémentation du protocole D-RPL, il donne d'abord une présentation globale de la structure du protocole RPL sous le simulateur Cooja. Il discute ensuite les modifications apportées à ce protocole dans le but de suppoter la métrique de délai de bout en bout.

Enfin, il introduit les modifications apportées au protocole et une description détaillé sur les fichiers touchés . Dans le chapitre suivant, nous allons décrire une série de simulations réalisées à l'aide du simulateur Cooja dans le but d'évaluer les performances du protocole D-RPL, et de le comparer avec RPL standard afin d'analyser les avantages et les inconvénients de cette extension.

Chapitre 5 : Simulation et Discussion des résultats

introduction

Dans les deux chapitres précédents, nous avons implémenté le protocole de routage D-RPL avec une QoS dans les réseaux LLNs. Il s'agit d'une extension du protocole RPL qui le rend capable d'assurer l'établissement des routes qui satisfont la contrainte de délai de bout en bout dans le but d'améliorer les performances des applications sensibles à cette métrique. Afin de valider et d'évaluer le mécanisme de routage avec QoS mis en place, l'implémentation des modifications apportées au protocole RPL a été réalisée sous le simulateur Cooja. Dans ce chapitre, nous allons présenter une série de simulations réalisées à l'aide du simulateur Cooja. Les objectifs de ces simulations sont les suivants :

- Evaluer les performances de la nouvelle implémentation selon certains paramètres de mesure,
- Comparer la nouvelle fonction objective avec OF0 du standard RPL ,
- Conclure l'impact des modifications apportées au protocole RPL.

5.1 Configuration du réseau

Nous concevons un exemple de réseau dans le simulateur Cooja contenant 40 nœuds clients et 1 nœud serveur en tant que racine du DODAG. Le scénario du réseau est illustré dans la figure 6.1. Le serveur utilise un exemple d'application udp-server.c alors que tous les autres

nœuds utilisent udp-client.c.

Afin d'introduire une perte dans le support sans fil, nous utilisons le Unit Disk Graph Medium de Cooja(UDGM) qui introduit une perte par rapport aux distances relatives aux nœuds dans le moyen radio. Les paramètres de la simulation et de son environnement sont indiqués dans le tableau 6.1

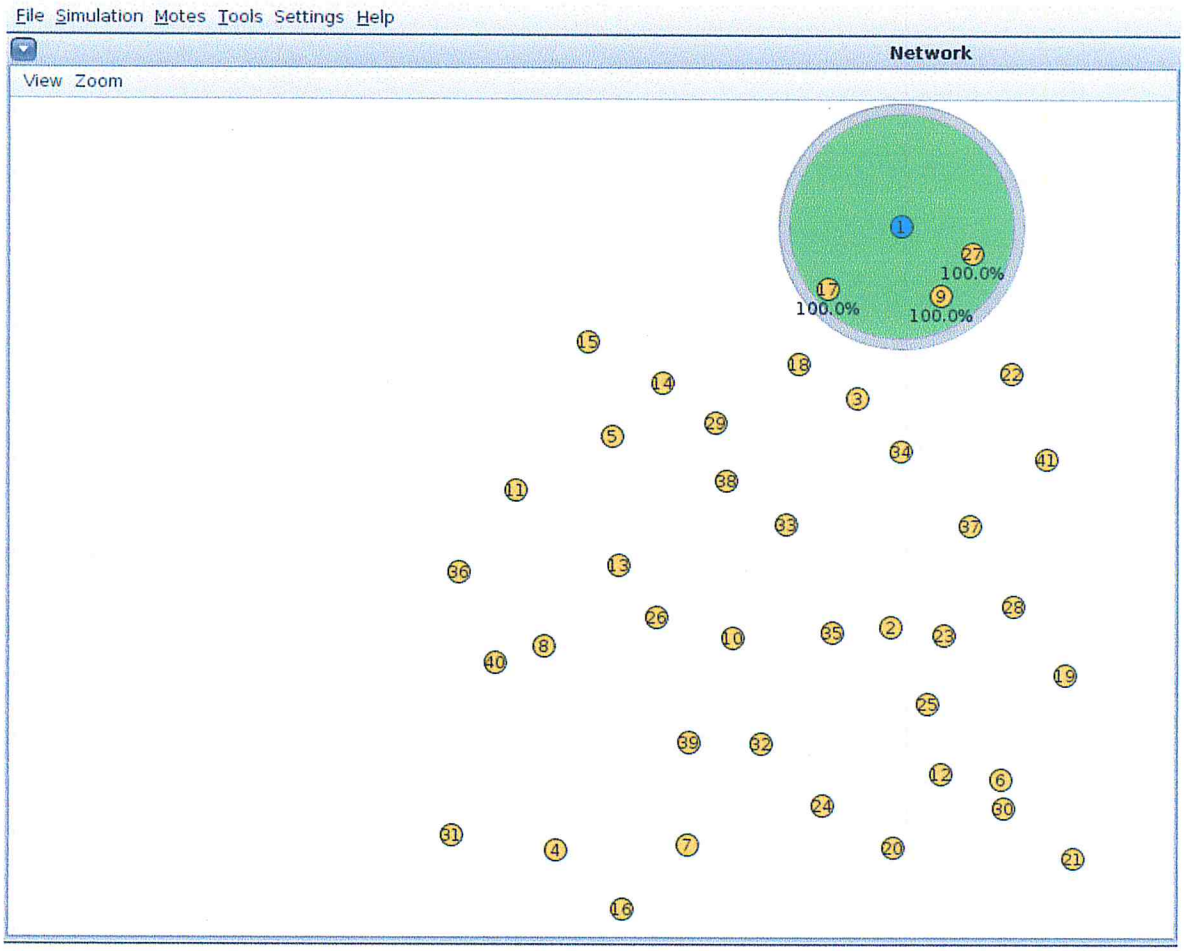


Figure 6.1: Configuration du réseau RPL

La figure 6.1 montre la configuration du réseau RPL avec 40 nœuds clients et 1 nœud racine dans le simulateur Cooja, Le nœud vert marqué comme 1 est la racine, tous les autres nœuds avec les cercles orange sont les nœuds clients et qui envoient des paquets au nœud

racine. Le plus grand cercle vert indique la zone de transmission et le cercle gris montre la zone de collision radio. Les chiffres en pourcentage indiquent le taux de réception.

Table 6.1: Paramètres de simulation

Paramètre	**Valeur**
Système d'exploitation	Contiki 2.7
Simulateur	Cooja
Temps de simulation	30 minutes
Nombre de simulation	20 simulations
Surface	300x300 mètres
Nombre de nœuds	1 sink et 40 clients
Fonction objective	OF0, D-RPL
Délai de démarrage	65 s
Modèle de rapport(Ratio model)	UDGM
Type d'application	UDP
Type de nœud	Cooja mote
Rang de transmission	50 mètres
Interval d'envoi de paquets de données	2 paquets/seconde
RX	30-100 %
Radio	CC2420

Comme le montre le tableau 6.1, le start delay est le délai de démarrage initial pour que l'application commence à transmettre ses messages au nœud racine. Ce temps de démarrage initial est le temps approximatif suffisant pour la convergence de réseau initiale. Cela garantit également que le paquet envoyé au serveur ne sera pas perdu en raison du manque de connectivité réseau. Par conséquent, une évaluation correcte peut être effectuée sur le nombre de paquets envoyés.

Le rapport de réception (RX) représente la perte de moyen radio.

La plage TX est réglée sur 50m et la portée des interférences à 55m.
Nous exécutons 20 simulations pour chaque fonction objective. Chaque simulation est exécutée pendant 30 min pour 40 nœuds.

5.2 Mesures de performance

Le réseau représenté dans la figure 6.1 à 40 nœuds et chaque nœud envoie un un paquet de données UDP ("Hello N", où N est le numéro de séquence de paquet) au serveur (sink) après chaque intervalle d'envoi et après un délai de démarrage initial (65 s). Le client imprime un message 'Hello N envoyé au serveur' dès qu'il envoie un paquet et cela nous permet de noter le temps d'envoi. De même, le serveur imprime un message 'Hello N reçu de Client M' et cela nous permet de noter le temps de réception du paquet sur le serveur. Les messages d'impression sont stockés dans un fichier journal à partir de la simulation. à partir du temps d'envoi et du temps de réception, nous calculons la latence pour tous les paquets.
Nous lisons le fichier journal ligne par ligne et notons l'identifiant du nœud, le numéro de paquet et le temps d'envoi dans un tableau. Lorsque nous lisons une ligne de 'Hello N reçue de Client M', nous recherchons dans le tableau l'ID de nœud et le numéro de paquet pour trouver le temps d'envoi de ce paquet. En cas de correspondance, nous calculons la latence et nous supprimons l'entrée correspondante dans le tableau. Toutes les entrées restantes dans le tableau indiquent les paquets perdus. La latence du réseau est calculée en utilisant l'équation suivante:

$$latence \quad totale = \sum_{k=1}^{n} temps \quad recu \quad - temps \quad d'envoi \tag{6.1}$$

Où n est le nombre total de paquets reçus avec succès. Les informations de synchronisation sont fournies par le simulateur Cooja . Pour calculer la latence moyenne, nous divisons la latence totale de l'équation 6.1 par le nombre total de paquets reçus. Le nombre total de paquets reçus est compté sur le nœud racine.

$$latence\ \ moyenne = latence\ \ totale\ \ /total\ \ des\ \ paquets\ \ recus \tag{6.2}$$

Pour calculer le taux moyen de livraison de paquets, nous mesurons le nombre de paquets envoyés de tous les nœuds à la racine et le divisons par le nombre de paquets reçus avec succès dans la racine.

$$PDR\ \ moyen = (Total\ \ des\ \ paquets\ \ recus/total\ \ des\ \ paquets\ \ envoyes) * 100 \tag{6.3}$$

Pour obtenir le temps de convergence dans un réseau RPL, nous déterminons l'heure du premier DIO envoyé depuis les nœuds clients et le dernier DIO a rejoint le DAG. Le temps de convergence est obtenu en soustrayant le premier temps envoyé par DIO à partir du moment de la dernière DIO jointe DAG.

$$Temps\ \ de\ \ convergence = dernier\ \ DIO\ \ rejoint\ \ DAG - Premier\ \ DIO\ envoye$$

5.3 Évaluation des performances

La fonction objective proposée a été testée et évaluée sur la moyenne de 20 simulations, et sur un scénario de test dont les nœuds sont placé sur une topologie aléatoire comme le montre la figure 6.2.

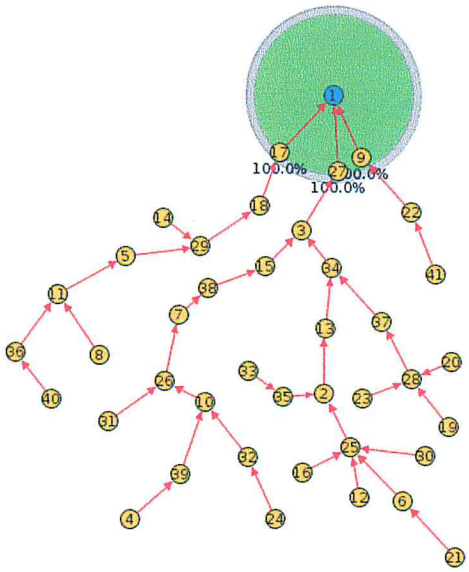


Figure 6.2: topologie aléatoire

La fonction objective D-RPL est comparée à la fonction objective par défaut: OF0. Dans toutes les simulations, nous avons utilisé un nœud racine (sink), un ensemble de nœuds clients (40 nœuds) qui envoient tous des paquets de données pour créer une situation congestionnée.

Nous avons calculé le délai de bout en bout pour un ensemble de nœuds et pour le réseau total pour D-RPL et nous l'avons comparé à la fonction objective OF0 du standard RPL, comme le montre la figures 6.3 et la figure 6.4. Les résultats montrent que D-RPL à un meilleur délai pour le réseau parraport à OF0.

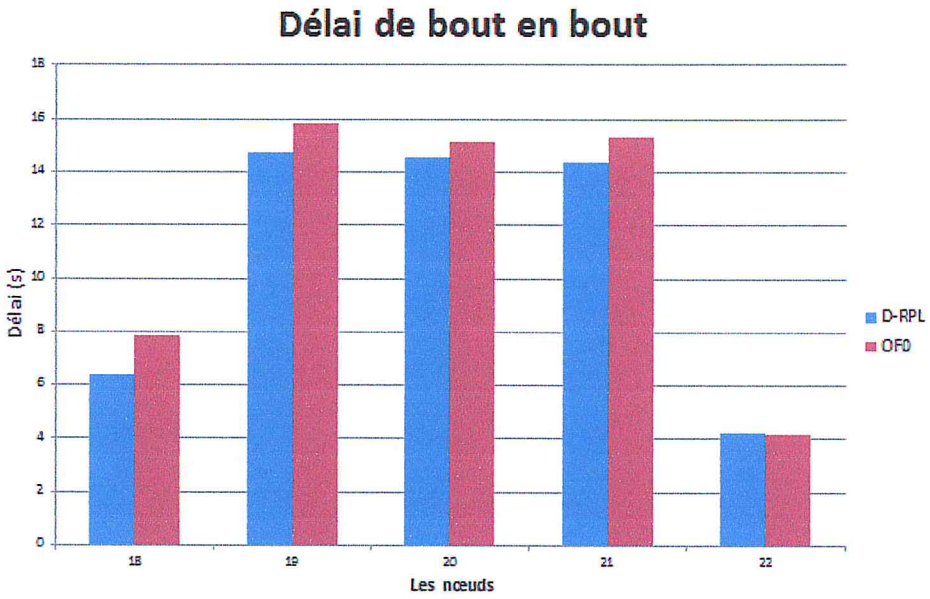


Figure 6.3: délai de bout en bout des nœuds.

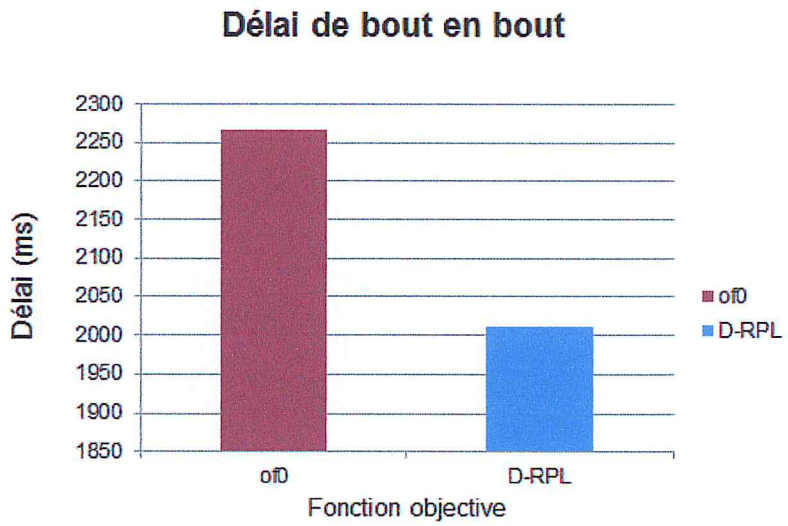


Figure 6.4: délai de bout en bout du réseau

Nous avons calculé le taux de réception de paquets dans le réseau pour D-RPL et nous l'avons comparé à la fonction objective OF0 du standard RPL, comme le montre la figure 6.5. Ce résultat montre que D-RPL à un taux de réception mieux que OF0 dans la majorité des cas, car D-RPL considère le délai comme une mesure dans sa fonction objective. Par

conséquent, D-RPL tente de transférer des paquets vers le sink à travers des nœuds moins congestionnés conduisant une perte réduite.

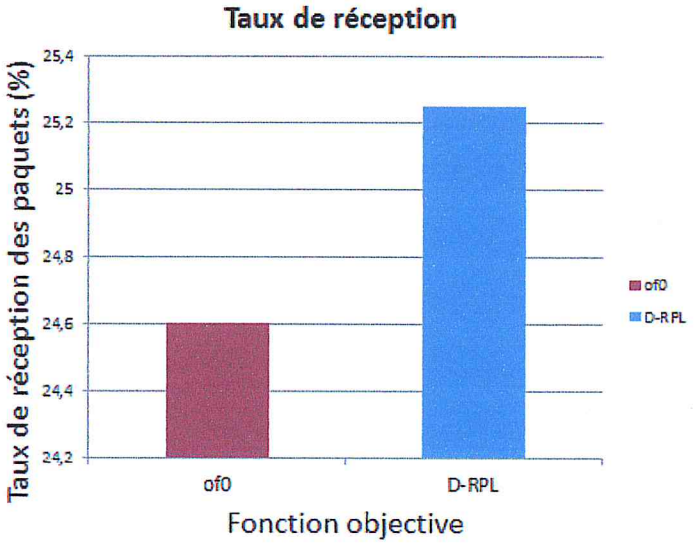


Figure 6.5: taux de réception du réseau.

Nous avons calculé aussi la gigue pour D-RPL et nous l'avons comparé à la fonction objective OF0 du standard RPL. La gigue est indépendante du délai de transmission. La gigue est une conséquence de congestions passagères sur le réseau, comme le montre la figure 6.6. Le résultat montre que D-RPL à une gigue acceptable dans la majorité des cas. où on remarque que dans les cas où OF0 est mieux on à de petits valeurs de différence donc on peut la considéré comme neglégable paraport á l'augmentation remarquable dans les cas où OF0 est mauvaise.

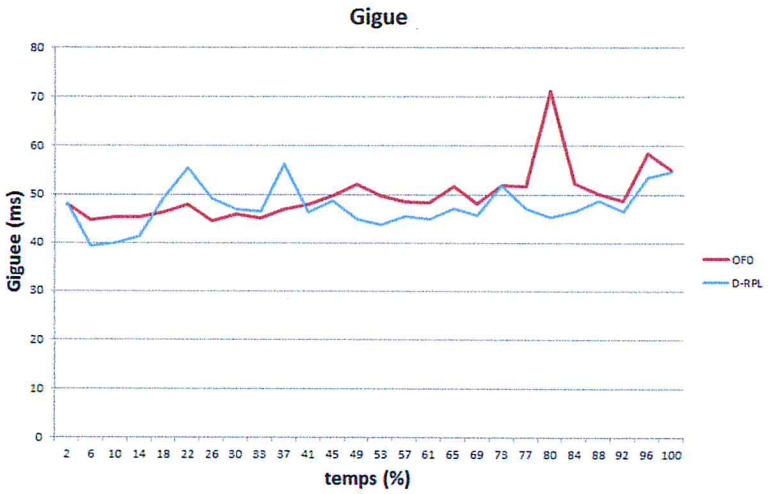


Figure 6.6: La gigue.

Dans l'ensemble, sur la base de ces résultats de simulation, il est clair que D-RPL améliore les performances en termes de délai et de taux réception de paquets, par rapport à OF0.

Conclusion

Dans ce chapitre, nous avons proposé une nouvelle métrique RPL importante à considérer lorsque la congestion se produit dans le réseau 6LoWPAN, la nouvelle fonction objective appelée D-RPL (Delay function for RPL) est proposée dans le protocole de routage RPL. La fonction objective proposée a été implémentée et testée dans Contiki 2.7 avec une topologie aléatoire et comparée avec la fonction objective par défaut OF0 du RPL standard.

Les résultats de la simulation montrent que D-RPL peut choisir le chemin le moins encombréd'un nœud feuille à un nœud racine en transférant des paquets à travers des nœuds moins congestionnés. Par conséquent, D-RPL améliore les performances du réseau en termes de délai et de taux de réception de paquets.

Conclusion générale

Les réseaux LLNs constituent un axe de recherche très fertile ainsi qu'un outil qui peut etre appliqué dans plusieurs domaines. Cependant, il reste encore de nombreux problèmes àrésoudre dans ce domaine pour un fonctionnement efficace de ces réseaux et afin de pouvoir les utiliser dans les conditions réelles. On ne peut pas abordé toutes ces problématiques en même temps, ce mémoire est centré sur celle de délai qui est une nécessité absolue à laquelle des solutions adéquates doivent être proposées et qui est causée par la perte de paquets de données et du fait que les objets d'IoT sont déployés dans des environnements complexes et hétérogenes et que leur batteries à faible puissance ne peuvent pas être rechargées, sachant que la communication entre les nœuds constitue l'opération la plus consommatrice d'énergie, donc on aura un gaspillage d'énergie qui est une ressource critique dans les réseaux LLNs, de ce fait, toute limitation de la communication entre les nœuds sera surement bénéfique.

Face aux problèmes de RPL dans LLNs, nous avons proposé une stratégie d'optimisation de routage qui est l'utilisation de délai dans la construction de l'arbre pour RPL, appelée DRPL. Les résultats de la simulation montrent que notre stratégie d'optimisation peut gérer la situation des liens instables et de la congestion du réseau, réduire le taux de perte de paquets et le délai de temps moyen du réseau et améliorer considérablement les performances des LLNs.

En utilisant la métrique D-RPL, le protocole RPL permet d'obtenir des performances en termes d'amélioration de retard. En outre, D-RPL parvient à avoir le meilleur pire en termes de retard. Tout cela, tout en ayant une topologie plus équilibrée en terme de choix de chemin.

Nous concluons également que l'implémentation d'un tel mécanisme de routage ayant une importante influence sur la stabilité du réseau. En effet, le routage avec contraints de délai peut apporter des améliorations à des applications exigeant des ressources de QoS telles que le trafic routier, les catastrophes. Il rend également l'implémentation de telles applications dans un réseau LLN plus efficace.

Ce travail nous a permis de se familiariser avec les réseaux LLNs, il nous a aussi fait découvrir une nouvelle plateforme de simulation adéquate qui est Cooja. En outre, dans ce

projet on a rencontré certaines difficultés dans la réalisation de ce projet tel que la limitation de la documentation sur le protocole RPL et la simulation sous Cooja. Ce qui nous laisse dire que pour arriver à une telle solution il faut bien sacrifier.

En guise de perspective, il serait intéressant de combiner et d'ajouter de nouvelles métriques pour contribuer à améliorer plus les performanes du protocoles RPL pour le rend encore plus efficace.

Annexes

6.1 Installation de Contiki

Contiki est une source de code ouvert et il marche directement sur les systèmes de Linux. Pour mettre en place contiki sur une machine, il y a deux façons :

- Première façon : télécharger une version Instant Contiki qui est situé dans le site http://sourceforge.net/projects/contiki/files/Instant Contiki/. En effet, c'est une version du système de l'exploitation Ubuntu, qui comprend le répertoire de contiki dedans. Cette façon d'installation est compatible avec une machine de Windows.
- Deuxième façon : utiliser l'outil Git pour synchroniser la dernière version du répertoire de contiki depuis le site https://github.com/contiki-os/contiki vers ton ordinateur. Cette façon d'installation est compatible avec une machine de Linux.

6.2 Le simulateur Cooja

Le simulateur Cooja est situé dans le dossier contiki-2.7 / tools / cooja. Il utilise une combinaison de code Java pour l'interface frontale et les émulateurs spécifiques à la plateforme pour effectuer les simulations.

6.3 Démarrage de Cooja

Vous pouvez démarrer le simulateur Cooja avec les commandes suivantes:

1. cd contiki-2.7/outils/cooja
2. ant run

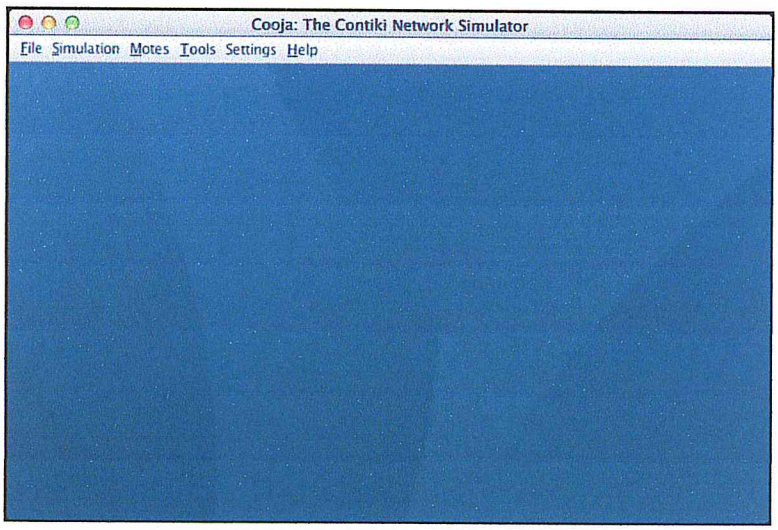


Figure 7.1: L'interface de bureau Cooja.

Vous verrez beaucoup de texte passer au moment où l'environnement Cooja est compilé, construit et démarré. Finalement, vous verrez maintenant l'interface de bureau Cooja, comme le montre la figure 7.2.

6.4 Création d'une simulation

Tout d'abord, vous devrez créer une nouvelle simulation. Pour ce faire, cliquez sur "Fichier> Nouvelle simulation". Cela présentera une boîte de dialogue, illustrée à la figure 2.

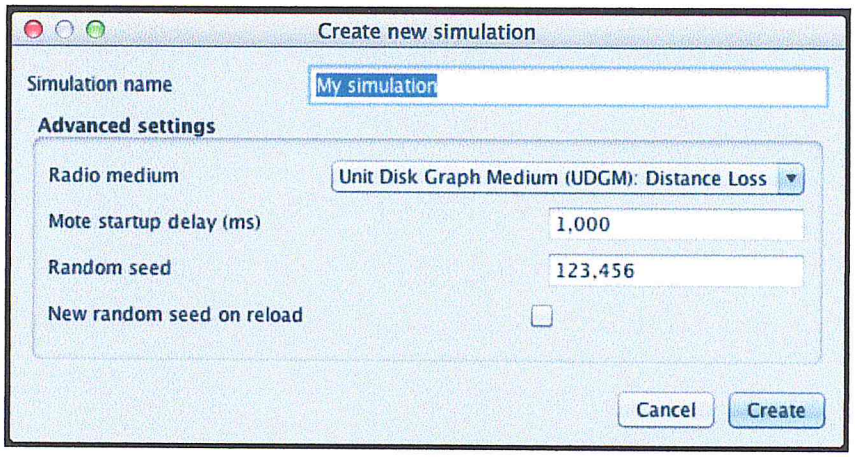


Figure 7.2: Créer une nouvelle interface de simulation.

Donnez à votre simulation un titre plus approprié, puis sélectionnez le support radio qui convient le mieux à votre type de simulation. Pour la plupart des simulations, Unit Disk Graph Medium (UDGM) est tout à fait approprié. Désactiver la simulation radio en sélectionnant "No Radio Traffic" ("Pas de trafic radio"), lorsqu'il n'est pas nécessaire, enregistre le temps CPU et rend la simulation plus rapide. Le démarrage aléatoire retarde le démarrage de chaque nœud simulée au hasard, de sorte qu'ils ne commencent pas tout à la fois exactement. La graine aléatoire (random seed) est une graine pour le générateur de nombres aléatoires. Vous pouvez cocher la case à la fin pour obtenir une graine aléatoire. Lorsque vous êtes prêt, cliquez sur "Créer".
Vous allez maintenant voir le bureau Cooja une fois de plus, mais avec l'environnement de simulation présenté, comme le montre la figure 7.3.

6.5 L'interface de simulation

L'interface de simulation, illustrée à la figure 7.3, se compose de cinq fenêtres.

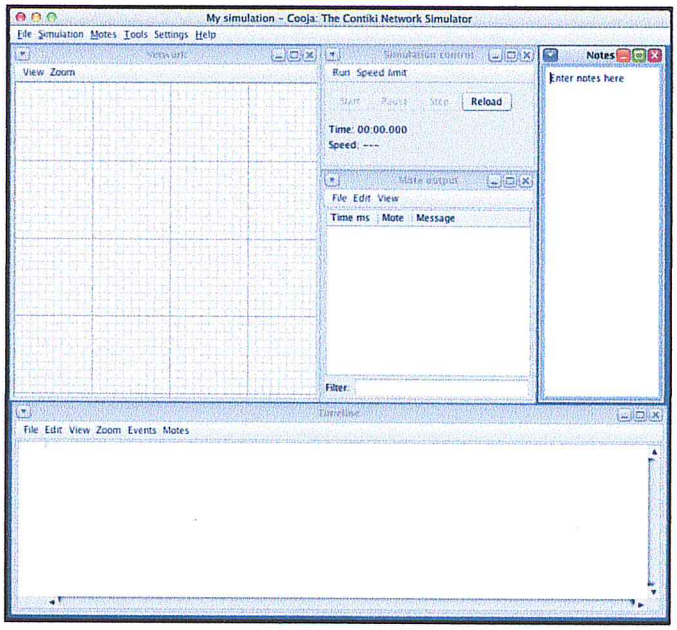


Figure 7.3: L'interface de simulation.

- La fenêtre Réseau affiche la disposition physique du réseau, c'est-à-dire que vous pourrez placer des nœuds ici et les déplacer, au besoin, afin de former la topologie et la mise en page qui vous intéressent.

- La fenêtre de contrôle de simulation vous permet de démarrer, Arrêtez et rechargez la simulation. Cela vous permet également de contrôler le taux de progression de la simulation.

- La fenêtre Mote Output montre toute sortie série générée par toutes les nœuds, c'est-à-dire la sortie de la commande printf. Vous pouvez filtrer la sortie affichée en fonction de la chaîne que vous entrez dans le champ "Filtre". Par exemple, si vous souhaitez filtrer la sortie de sorte qu'elle affiche uniquement la sortie de nœud 2, vous pouvez entrer "ID:2" dans ce champ.

- La fenêtre Timeline affiche les événements qui se produisent sur chaque nœud au cours de la chronologie de la simulation. Ces événements peuvent être le trafic radio, l'activité LED ou toute autre chose.

- La fenêtre Notes peut être utilisée pour prendre des notes temporaires dans la simulation.

6.6 Définition des types de nœuds

La prochaine étape consiste à définir les types de nœuds. Dans Cooja, vous créez une liste de types de nœud, définissez leurs paramètres et attribuez leur code de programme / binaire. Vous alors, en tant que tâche séparée, créez des instances pour simuler. Nous abordons ici la première partie: créer les types de noeuds.

Cliquez sur "Motes > Add Motes > Create New Mote Type > cooja mote.". Donnez au noeud une description utile. voir la figure 7.4.

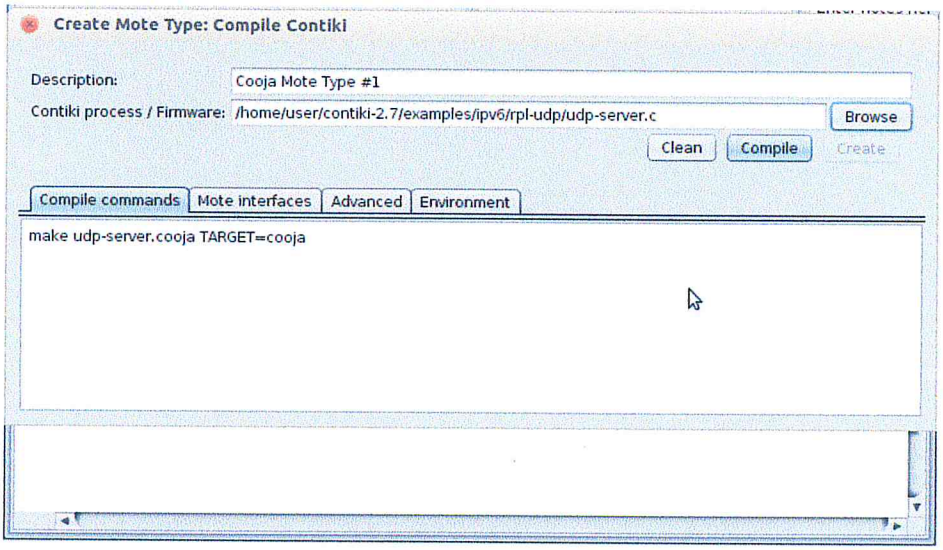


Figure 7.4: L'interface creer le type de nœud

Dans le champ "Contiki Process / Firmware", vous devez spécifier votre fichier source (le fichier .c) ou le fichier binaire (le fichier .sky). Si vous spécifiez le fichier binaire, vous ne verrez aucune instruction de compilation (les fichiers binaires sont le résultat de la compilation - il n'est pas nécessaire de compiler). Si vous spécifiez le code source, le champ de commandes

de compilation devient actif et vous pouvez spécifier des instructions spécifiques pour la compilation. pour nous, nous allons utiliser le bouton Browse (Parcourir) pour sélectionner le fichier udp-server.c, apres nous allons cliquer sur Compiler pour pouvoir créer le type de nœud. Vous pouvez voir la sortie / résultat de la compilation dans l'onglet de sortie de compilation. En cas de succès, le bouton Créer est disponible pour creer vos nœuds. Dans le cas, où vous utilisez un binaire pré-compilé (le fichier .sky), le bouton Créer est déjà disponible sans compiler. De même, vous pouvez créer le client UDP en utilisant le fichier udp-client.c.

6.7 Ajout de nœuds (Motes)

Maintenant que vous avez créé les types de nœuds udp-server et udp-client, nous pouvons commencer à ajouter des nœuds physiques à la simulation. Le dialogue "Add motes" vous permet de définir le nombre et la configuration de ces nouvelles actions. Vous pouvez accéder a cela en cliquant sur "Motes > Add Motes". Ajoutez un nœud du type udp-server et cinq nœuds de type udp-client par exemple. Vous pouvez utiliser le positionnement aléatoire pour ajouter les nœuds.

Une fois que vous faites cela, vous constaterez qu'un total de six nœuds placés au hasard apparaîtront dans la fenêtre Réseau. Un arrangement aléatoire possible peut être vu àla figure 7.7 (a). Parmi ceux-ci, le nœud 1 est le serveur et le reste sont des nœuds qui exécuteront le code du serveur UDP. Vous pouvez interagir avec chacune des nœuds en cliquant et en les faisant glisser.

Vous pouvez réorganiser la disposition des nœuds en les faisant glisser autour, de sorte qu'ils formeront une topologie qui vous intéresse. Lorsque vous cliquez sur un nœud, il affiche egalement l'environnement radio en couleur verte et grise. Le cercle vert représente la zone dans laquelle le signal a été reçu avec succès par d'autres nœuds et le gris représente la zone où l'interférence radio du nœud actuel. Un exemple de cet environnement radio et une nouvelle mise en page telle que seul un maximum de deux nœuds se situent dans la portée de chaque nœud, peut être vu à la figure 7.7(b).

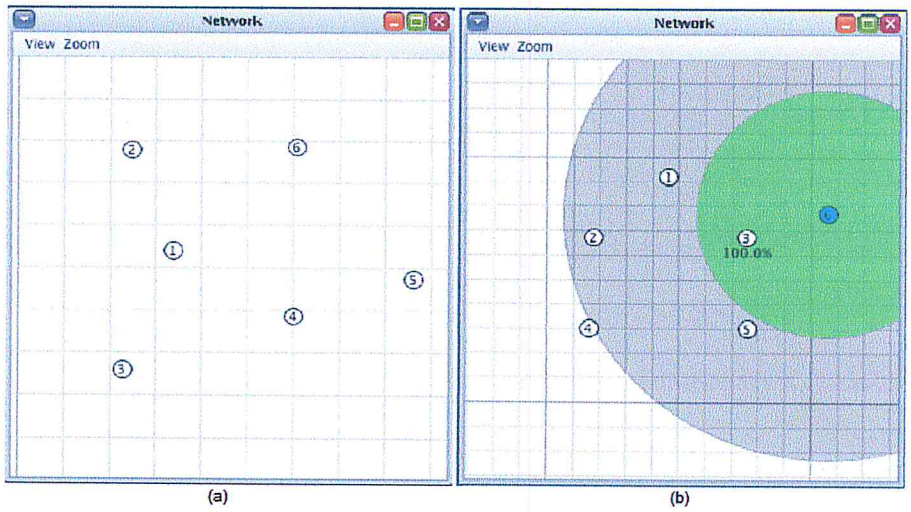


Figure 7.5: La fenêtre du réseau.

Conclusion

Vous savez maintenant comment travailler avec le simulateur Contiki Cooja.

Bibliographie

[1] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):394, 2002.

[2] Kevin Ashton. That internet of things thing. *RFiD Journal*, 22(7), 2009.

[3] Yibo Chen, Jean-Pierre Chanet, Kun-Mean Hou, Hongling Shi, and Gil de Sousa. A scalable context-aware objective function (scaof) of routing protocol for agricultural low-power and lossy networks (rpal). *Sensors*, 15(8):19507–19540, 2015.

[4] Alex Conta and Mukesh Gupta. Internet control message protocol (icmpv6) for the internet protocol version 6 (ipv6) specification. 2006.

[5] Olfa Gaddour, Anis Koubaa, Nouha Baccour, and Mohamed Abid. Of-fl: Qos-aware fuzzy logic objective function for the rpl routing protocol. In *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), 2014 12th International Symposium on*, pages 365–372. IEEE, 2014.

[6] Mohamed Ghazi Amor, Anis Koubâa, Eduardo Tovar, and Mohamed Khalgui. Cyberof: An adaptive cyber-physical objective function for smart cities applications. In *28th Euromicro Conference on Real-Time Systems*, 2016.

[7] Amitabha Ghosh and Sajal K Das. Coverage and connectivity issues in wireless sensor networks: A survey. *Pervasive and Mobile Computing*, 4(3):303–334, 2008.

[8] Omprakash Gnawali. The minimum rank with hysteresis objective function. 2012.

[9] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1649, 2013.

[10] D INFSO. 4networked enterprise&rfid infso g. 2micro&nano systems in cooperation with the rfid working group of the etp eposs. internet of things in2020: A roadmap for the future [ol], 4.

[11] Oana Iova, Fabrice Theoleyre, and Thomas Noel. Improving the network lifetime with energy-balancing routing: Application to rpl. In *Wireless and Mobile Networking Conference (WMNC), 2014 7th IFIP*, pages 1–8. IEEE, 2014.

[12] K Kim, Park Daniel, and N Montenegro. Kushalnagar, 6lowpan ad hoc on-demand distance vector routing (load). *IETF, draft-daniel-6lowpan-load-adhoc-routing-03. txt*, 2007.

[13] Gabriel Montenegro, Nandakishore Kushalnagar, Jonathan Hui, and David Culler. Transmission of ipv6 packets over ieee 802.15. 4 networks. Technical report, 2007.

[14] Tarcisio Bruno Oliveira, Pedro Henrique Gomes, Danielo G Gomes, and Bhaskar Krishnamachari. Alabamo: A load balancing model for rpl.

[15] Charles Perkins, Elizabeth Belding-Royer, and Samir Das. Ad hoc on-demand distance vector (aodv) routing. Technical report, 2003.

[16] Adam Petcher. Qos in wireless data networks. *Washington University in St. Louis. Department of Computer Science & Engineering. Esitelmä (27.2. 2006 & 1.3. 2006) kurssilla CSE574S: Advanced Topics in Networking: Wireless and Mobile Networking (Spring 2006)*, pages 4–5, 2006.

[17] Tausifa Jan Saleem. A detailed study of routing in internet of things. *network*, 5(3), 2016.

[18] Anuj Sehgal. Using the contiki cooja simulator. *Computer Science, Jacobs University Bremen Campus Ring*, 1:28759, 2013.

[19] Bassam Sharkawy, Ahmed Khattab, and Khaled MF Elsayed. Fault-tolerant rpl through context awareness. In *Internet of Things (WF-IoT), 2014 IEEE World Forum on*, pages 437–441. IEEE, 2014.

[20] Harald Sundmaeker, Patrick Guillemin, Peter Friess, and Sylvie Woelfflé. Vision and challenges for realising the internet of things. *Cluster of European Research Projects on the Internet of Things, European Commision*, 2010.

[21] Kazuo Takaragi, Mitsuo Usami, Ryo Imura, Rei Itsuki, and Tsuneo Satoh. An ultra small individual recognition security chip. *IEEE micro*, 21(6):43–49, 2001.

[22] Pascal Thubert. Objective function zero for the routing protocol for low-power and lossy networks (rpl). 2012.

[23] Jean-Philippe Vasseur, Mijeom Kim, Kris Pister, Nicolas Dejean, and Dominique Barthel. Routing metrics used for path calculation in low-power and lossy networks. Technical report, 2012.

[24] Evan Welbourne, Leilani Battle, Garret Cole, Kayla Gould, Kyle Rector, Samuel Raymer, Magdalena Balazinska, and Gaetano Borriello. Building the internet of things using rfid: the rfid ecosystem experience. *IEEE Internet Computing*, 13(3):48–55, 2009.

[25] Tim Winter. Rpl: Ipv6 routing protocol for low-power and lossy networks. 2012.

جامعة البلقاء التطبيقية
مكتبة الكلية
كلية العلوم