République Algérienne Démocratique Et Populaire
Université Saad Dahleb Blida1
Faculté des Sciences
Département d'informatique

Thème

Proposition et Implémentation d'un Protocole de Routage basé sur les SDNs pour l'Internet des Objets

En vue d'obtention du diplôme de Master
Domaine : MI
Filière : Informatique
Spécialité : Système informatique et réseaux

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

Rapport présenté par :
Fares BOURAS
Rania BOUZEGHLANE

Année universitaire : 2020/2021

Remerciement

En premier lieu, nous tenons à remercier Dieu l'absolu et le tout puissant, qui nous a aidé à faire ce modeste travail et qui nous a donné la force afin de continuer jusqu' au bout.

Après, Nous tenons également à remercier notre promoteur : ***Mr. Mohamed OULD-KHAOUA*** *et notre encadreur :* ***Mr. Nadir BOUCHAMA*** *pour leurs précieux conseils au cours des différentes étapes du travail.*

Nous adressons aussi nos remerciements aux membres du jury qui ont accepté d'examiner notre travail et nous vous sommes très reconnaissants de bien vouloir porter intérêt à ce travail.

Enfin, un grand merci à toutes les personnes qui ont participé à la réalisation de ce travail.

Dédicaces

Nous dédions notre travail :

À notre famille et à nos chers parents pour leur soutien tout au long de notre parcours académique.

A Yacine, Mehdi, Iyad, Samir, Chérif, Amira, Meriem, Amel et Rania.

À tous ceux qui nous sont chers et qui nous aiment.

Fares/Rania

Résumé

A l'opposé des réseaux LLN classiques qui sont une solution distribuée, les réseaux de type SDN utilisent un système centralisé, qui consiste à mettre en place un contrôleur distant qui gère tout le réseau en ce qui est routage, sécurité, qualité de service etc.

Dans notre travail, nous allons implémenter et évaluer les performances du protocole SDN-WISE qui est basé sur les SDNs et adéquat pour l'internet des objets en le comparant avec le protocole RPL (*Routing Protocol for Low power and lossy networks*).

Nous avons choisi le SDN-WISE pour son efficacité énergétique, sa robustesse, son évolutivité et son adaptabilité.

Les résultats obtenus des simulations faites dans des environnements différents démontrent que notre solution choisie (SDN-WISE) est très performante en termes de consommation d'énergie et en taux de réception de paquet dans par rapport à RPL.
Mots clés: Internet des Objets, réseau LLN, routage, Software Defined Networking, RPL, évaluation de performances.

Abstract

Unlike conventional LLN networks which are a distributed solution, SDN type networks use a centralized system, which consists of setting up a remote controller which manages the entire network in terms of routing, security, quality of service, etc. .

In our work, we will implement and evaluate the performance of the SDN-WISE protocol which is based on SDNs and suitable for the Internet of Things by comparing it with the RPL protocol (Routing Protocol for Low power and lossy networks).

We chose SDN-WISE for its energy efficiency, robustness, scalability and adaptability.

The results obtained from simulations carried out in different environments demonstrate that our chosen solution (SDN-WISE) is very efficient in terms of power consumption and packet reception rate in compared to RPL.
Keywords: Internet of Things, LLN network, routing, Software Defined Networking, RPL, performance evaluation.

الملخص

على عكس الشبكات منخفضة الطاقة والفقدان التي تعتبر حلاً موزعًا تستخدم شبكات معرفة بالبرمجيات نظاما مركزيا يتكون من إعداد وحدة تحكم عن بعد تدير الشبكة بأكملها من حيث التوجيه و الأمان و جودة الخدمة, وما إلى ذلك. الهدف من هذا العمل هو تنفيذ و تقييم أداء بروتوكول توجيه قائم على مناسب لانترنت الأشياء من خلال مقارنته ببروتوكول آخر في شبكة تقليدية.

توضح النتائج التي تم الحصول عليها أن الحل المختار فعال للغاية من حيث استهلاك الطاقة و معدل استقبال الحزم في بيئات مختلفة.
الكلمات المفتاحية: انترنت الأشياء, الشبكات منخفضة الطاقة و الفقدان, الشبكات المعرفة بالبرمجيات, تقييم الأداء.

Table des matières

Liste des figures

Liste des tableaux

Liste des abréviations

IOT: Internet of Things.
SDN: Software Defined Network.
LLN: Low power and Lossy Network.
RPL: Routing Protocol for Low-power and lossy networks.
DODAG: Destination Oriented Directed Acyclic Graph.
DIO: Destination Information Object.
DAO: Destination Advertissement Object.
DAO-ACK: DAO-Acknowledgment.
DIS: DODAG Information Sollicitation.
ETX: Expected Trasmission count.
RSSI: Received Signal Strength Indicator.
PRR: Packet Reception Rate.
RWM: Random Waypoint Model.
WSN: Wireless Sensor Network.
PDR: Packet Delivery Ratio.
IETF: Internet Engineering Task.
IPv6: Internet Protocol version 6.
OF: Objective Function.
MRHOF: Minimum Rank Hysteresis Objective Function.
QoS: Quality of Service.
LQI: Link Quality Indicator.
INPP: In-Networking Packet Processing.
TD: Topology Discovery.
MANET: Mobile Ad hoc NETwork.
VANET: Vehicular Ad hoc NETwork.
ROLL: Routing Over LLNs.
RAN: Radio Access Network.
SOF : Sensor OpenFlow.

Introduction générale

L'internet des Objets (IoT) peut être défini de plusieurs manières. Selon la recommandation ITU-T Y.2060 de l'ITU, l'Internet des Objets est une « infrastructure mondiale pour la société de l'information, qui permet de disposer de services évolués en interconnectant des objets (physiques ou virtuels) grâce aux technologies de l' information et de la communication interopérables existantes ou en évolution ». L'internet des Objets à des applications dans les grilles intelligentes (smart grids), la domotique (smart home), l'agriculture de précision, la e-santé (e-health), le transport intelligent (smart transportation), etc.

Un des problèmes majeurs dans l'IoT est le routage. En effet, les travaux menés par le groupe de travail ROLL (*Routing Over LLNs*) de l'IETF ont montré que les protocoles de routage dédiés aux réseaux ad hoc (AODV, OLSR, DSR, TBRPF, DYMO) étaient inadaptés aux réseaux LLNs (Low power and Lossy Networks). De ce fait, ROLL a essayé de proposer de nouveaux protocoles de routage dédiés aux LLNs, dont RPL (*Routing Protocol for LLNs*).

Les réseaux définis par logiciels ou SDNs (*Software Defined Networks*) sont un nouveau paradigme qui décrit une architecture réseau dont le plan de contrôle (control plan) est totalement découplé du plan de données (data plan). Cette approche peut être utilisée pour améliorer la qualité de service, la scalabilité, l'équilibrage de charge, la sécurité, etc. D'autre part, elle peut être couplée avec les nouvelles technologies telles que le Big Data, l'IoT, le cloud computing, et le machine learning.

Dans cette perspective les chercheurs ont développé plusieur solutions SDN adapté pour l'internet des objets, selon le milieu ou bien l'environnement chaque solution a ses avantages et ses inconvénients, parmi les solutions proposés SDN-WISE a été l'un des meilleurs protocoles actif dans l'IoT, avec l'architecture du contrôleur centralisé il donne plusieurs avantage au réseau comme la scalabilité, la reconfiguration du réseau, moins de consommation d'énergie etc. Plusieurs tests ont été faits sur ce protocole, les chercheurs on fait une analyse comparative entre SDN-WISE et le protocole RPL [31] selon leurs paramètres d'évaluation et leur environnement proposé ils ont eu de différents résultats et au final ils se sont arrivés à un résultat qui favorise l'utilisation de RPL que SDN-WISE.

Dans notre travail on va prouver que SDN-WISE et mieux que RPL en ce qui concerne les métriques qu'on a pris en considération et aussi dans un environnement Mobile contrairement à l'étude qui a été faite dans un environnement fixe, ce qui va démontrer que SDN-WISE est une très bonne solution pour l'IoT.

Bibliographie recommandée :

Bekri, Wiem, Rihab Jmal, and Lamia Chaari Fourati. "Internet of things management based on software defined networking: a survey. " International Journal of Wireless Information Networks 27 (2020): 385-410.

Baddeley, Michael. Software Defined Networking for the Industrial Internet of Things. Diss.University of Bristol, 2020.

Theodorou, Tryfon, and Lefteris Mamatas. "SD-MIoT: A Software-Defined Networking Solution for Mobile Internet of Things. " IEEE Internet of Things Journal (2020).

Baddeley, Michael, et al. "Atomic-SDN: Is synchronous flooding the solution to software-defined networking in IoT." IEEE Access 7 (2019): 96019-96034.

Choukri, Ihssane, Mohammed Ouzzif, and Khalid Bouragba. "Software Defined Networking (SDN): Etat de L'art." Colloque sur les Objets et systèmes Connectés. 2019.

Tsapardakis, Eleftherios, et al. "Performance evaluation of SDN and RPL in wireless sensor networks. " 2018 Global Information Infrastructure and Networking Symposium (GIIS). IEEE, 2018.

Bera, Samaresh, Sudip Misra, and Athanasios V. Vasilakos. "Software-defined networking for internet of things: A survey. " IEEE Internet of Things Journal 4.6 (2017): 1994-2008.

L'organisation du mémoire est comme suit:

Chapitre 1: Une introduction ou bien un Background sur l'internet des objets, le protocole RPL et l'architecture SDN de façon générale.

Chapitre 2: Etat de l'art sur les solutions SDN dans l'IoT, le chapitre site et décrit plusieurs architectures proposées ainsi que les différents avantages et défis du SDN.

Chapitre 3: Ce chapitre détaille l'architecture de SDN-WISE, le format des paquets, table etc. Et aussi le mécanisme du protocole.

Chapitre 4: Evaluation des performances et qui expose les résultats de la simulation avec leur discussion.

Chapitre 1
Introduction à l'internet des objets

Introduction

L'Internet des objets (*IoT*) est un paradigme de communication récent qui envisage un futur proche, dans lequel les objets de la vie quotidienne seront équipés de microcontrôleurs, d'émetteurs-récepteurs pour la communication numérique et un ensemble de protocoles appropriés qui les rendront capables de communiquer entre eux et avec les utilisateurs, pour devenir une partie intégrante de l'Internet [1], dans ce scénario complexe, l'application du paradigme IoT à un contexte urbain est d'un intérêt particulier, car il répond à la forte poussée de nombreux gouvernements nationaux à adopter des solutions dans la gestion des affaires publiques et l'amélioration des services de la vie quotidienne, réalisant ainsi le concept de la ville intelligente[2].

La connectivité des objets se base sur une architecture réseau bien définie, il ne faut pas confondre le principe des réseaux capteurs sans fils WSN (*Wireless Sensor Network*) avec le système de l'internet des objets.

Dans un système IoT, tous les capteurs envoient directement leurs informations sur Internet. Par exemple, un capteur peut être utilisé pour surveiller la température d'une masse d'eau. Dans ce cas, les données seront immédiatement ou périodiquement envoyées directement sur Internet [3], où un serveur peut traiter les données.

Inversement, dans un WSN, il n'y a pas de connexion directe à Internet. Au lieu de cela, les différents capteurs se connectent à une sorte de routeur ou de nœud central. Une personne peut ensuite acheminer les données du routeur ou du nœud central comme elle l'entend [3]. Cela étant dit, un système IoT peut utiliser un réseau de capteurs sans fil en communiquant avec son routeur pour collecter des données.

1. Les réseaux LLNs

Un réseau de capteurs se compose de plusieurs nœuds, les problèmes majeurs sont groupés autour de la consommation d'énergie et le taux de livraison des paquets PDR (*Packet Delivery Ratio*),d'où vient l'appellation LLNs (*Low power and Lossy Networks*) Réseau à faible puissance et avec perte, généralement composé de nombreux périphériques intégrés avec une puissance, une mémoire et des ressources de traitement limitées interconnectés par une variété de liens, comme Le ZigBee 802.15.4 ou le Low Power Wi-Fi ,Il existe beaucoup de domaines d'application pour les LLN, y compris l'automatisation des bâtiments (chauffage, ventilation et climatisation (HVAC) [4], éclairage, contrôle d'accès, incendie), la maison connectée, les soins de santé, la surveillance environnementale, les réseaux de capteurs urbains, gestion de l'énergie etc.

2. Le Routage Dans les LLNs

Les LLNs sont composés de quelques dizaines à des milliers de routeurs et prennent en charge le trafic point à point (entre les périphériques à l'intérieur du LLN), le trafic point à multipoint (d'un point de contrôle central à un sous-ensemble de périphériques à l'intérieur du LLN) et du trafic multipoint à point [5] (des appareils à l'intérieur du LLN vers un point de contrôle central). Des protocoles de routage sont nécessaires pour établir et maintenir une connectivité multi-hop dans les LLN, [6] pour les situations où il est impossible de fournir un déploiement de réseau de capteurs de telle sorte que tous les périphériques nécessitent une communication entre eux. Pour garantir un bon acheminement de données et satisfaire les recommandations de ce type de réseau, il fallait développer un protocole économe, qui ne consomme pas beaucoup d'énergie et qui garantit un bon routage dans une durée de temps optimisée, adapté pour les réseaux à faible puissance. IETF (*The Internet Engineering Task Force*) a développé RPL (*Routing Protocol for Low-Power and Lossy networks*) comme protocole de routage pour les LLNs [7] et l'ont standardisé dans la RFC 6550.

Le protocol RPL

1. Contexte

RPL est un protocole de routage IPv6 conçu par le groupe de travail IETF ROLL pour les réseaux à faible puissance et avec perte (*LLN*), il fonctionne sur la norme IEEE 802.15.4, il utilise 6LoWPAN comme couche d'adaptation [7].

2. 6LoWPAN

La transmission des paquet de protocole Internet version 6 (*IPv6*) sur le réseau personnel sans fil à faible consommation (*LoWPAN*) a été normalisé par l' IETF en tant que protocole 6LoWPAN [8], Il fournit au nœud du réseau de capteurs sans fil (*WSN*) des capacités de communication IP en plaçant une couche d'adaptation au-dessus de la couche de liaison 802.15.4[9].

6LoWPAN est un protocole adapté pour un réseau de communication simple à faible coût qui permet une connectivité sans fil avec puissance limitée et exigences de débit assouplies car il fournit un réseau IPv6 au-dessus des réseaux IEEE 802.15.4 comme le montre la figure 1 [8,14]. Adapté pour les appareils compatibles avec la norme IEEE 802.15.4 et caractérisés par une courte portée, faible débit binaire, faible puissance, faible utilisation de la mémoire et faible coût.

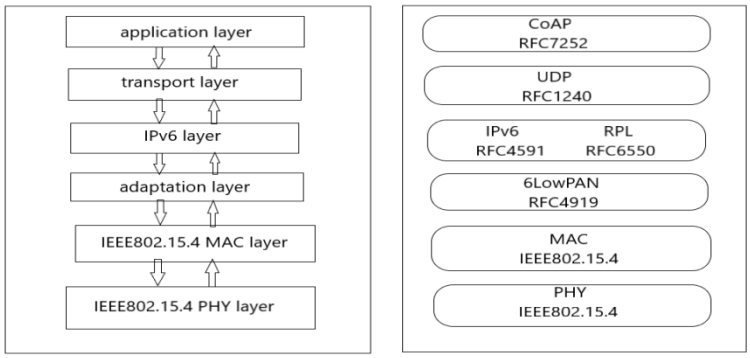

Figure 1.1- Couches du RPL [8,14]

Un système 6LoWPAN est un réseau maillé sans fil de faible puissance où chaque nœud est identifié par sa propre adresse IPv6 qui lui permet de se connecter directement à Internet comme le montre la figure 2 [10].

3. Les messages de contrôle dans RPL

RPL construit la topologie du réseau en se basant sur un graphe acyclique dirigé (DAG) sans arêtes sortantes afin qu'aucun cycle ou boucle ne puisse exister. Chaque DAG est acheminé vers une ou plusieurs racines de DAG formant un DAG orienté destination (DODAG) et chaque DODAG a son propre DODAG-ID, qui lui est attribué et considéré comme un identifiant [11].

Dans ce protocole, il existe 4 types de messages, chacun est utilisé pour une fonction bien spécifique comme le montre la figure 1.2 [11].

a. DIO (*DODAG information Object*)

RPL maintient la connectivité à l'aide d'un certain nombre de messages de contrôle, le DODAG information Object (DIO) qui transporte des informations y compris DODAG-ID, le rang pour permettre aux autres nœuds de découvrir le DODAG [12] et le numéro de version du DODAG permettant de savoir si les informations reçues sont récentes ou obsolètes. Le rang d'un nœud correspond à son emplacement dans le graphe par rapport à la racine. La valeur du rang augmente toujours en descendant dans le graphe. C'est donc la racine qui a le plus petit rang dans le graphe [13].

b. DAO (*Destination Advertisement Object*)

Le DAO contient l'ID instancié RPL qui a été appris du DIO et qui est envoyé du nœud enfant au nœud parent ou à la racine DODAG qui représente l'extrémité du réseau. Il est utilisé pour propager les informations de route inverse pour enregistrer les nœuds visités le long du chemin ascendant

c. DAO-Ack (*DAO-Acknowledgment*)

Il est utilisé par la destination en réponse à un message DAO reçu pour acquitter ce dernier. En cas de non réception du DAO-Ack par la source, celle-ci peut réémettre le DAO initial [14].

d. DIS (*The DODAG Information Sollicitation*)

Ce type de message est généralement utilisé pour demander un DIO à un nœud RPL afin de découvrir le réseau et pour gérer tous types de Congestion. Un nœud qui reçoit un DIS répond à l'initiateur par une monodiffusion de paquet DIO.

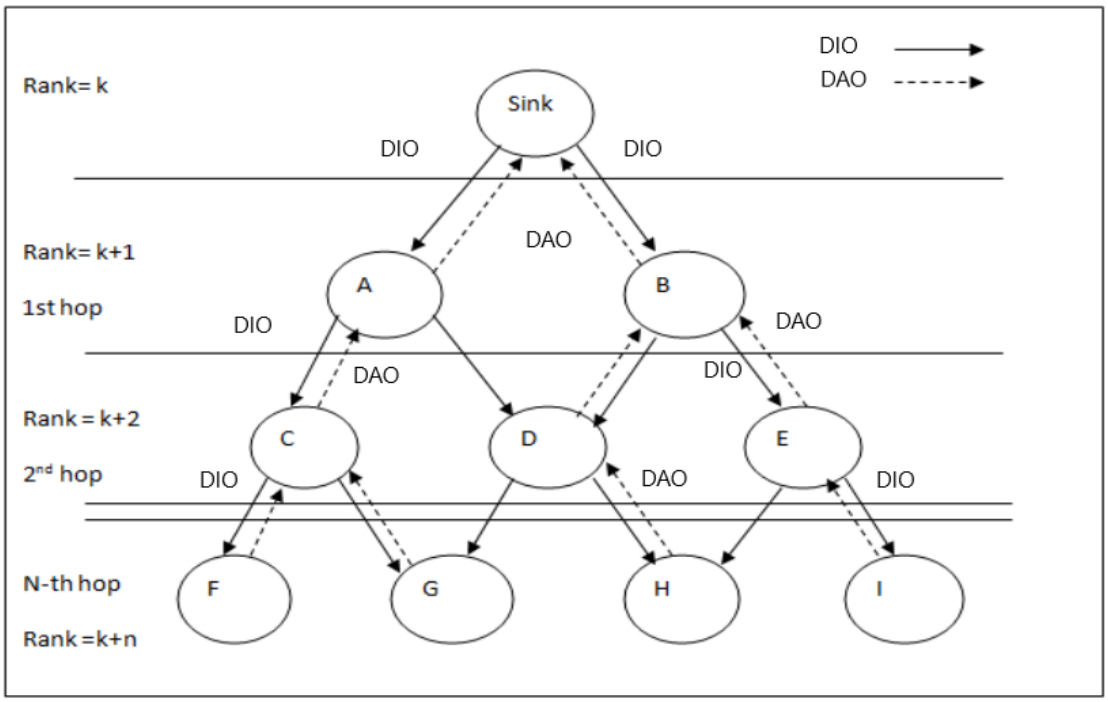

Figure 1.2- Le DOODAG [10]

4. Fonctionnement du Protocol RPL

a. Maintenance des liens

Afin de ne pas consommer beaucoup d'énergie par l'envoi des messages DIO, RPL utilise l'algorithme du Trickle qui est standardisé dans la RFC 6206 [14]. L'idée de base de cet algorithme est d'adapter dynamiquement la fréquence d'envoi des DIO en fonction de la qualité des informations à transmettre : plus la topologie est stable, moins il est nécessaire d'envoyer les informations de contrôle.

Au début, la fréquence d'envoi des DIO est faite selon un intervalle minimum, défini par le paramètre de configuration **"I-min"**. Cet intervalle double régulièrement à l'expiration du minuteur d'envoi, jusqu'à une valeur maximale : **"I-**

max" à laquelle elle se stabilise. Dès qu'une incohérence est constatée ou qu'une nouvelle information est disponible, le minuteur de l'algorithme est réinitialisé.

La nature non fiable des transmissions dans les LLN dont les pertes éventuelles de paquets peut conduire à un état d'incohérence qui est défini par **"un état instable"**, lorsqu'un nœud reçoit d'un voisin des informations obsolètes (numéro de version d'instance antérieur) [14]. Il doit alors réinitialiser son minuteur Trickle pour diffuser immédiatement l'information actuelle dont il dispose.

b. Rang et gestion des boucles

Pour éviter la formation de boucles de routage, RPL qui est un routage à vecteur de distance, utilise le rang. Ce dernier indique pour chaque nœud, sa position relative vis-à-vis de la racine comparée aux nœuds du voisinage. Il s'agit d'une grandeur scalaire encore appelée rang dans le standard.

La façon dont le rang est calculé par une certaine fonction n'est pas précisée, mais il doit représenter des propriétés génériques quelle que soit la fonction utilisée, il doit être en particulier décroissant lorsqu'on progresse vers la racine.

RPL offre un mécanisme pour détecter l'existence d'une boucle pendant la transmission des données. Une nouvelle option [14] est incorporée dans les paquets IPv6 qui transitent par un réseau utilisant RPL.

Elle prévoit un bit qui peut être positionné pour indiquer le sens de progression du paquet (0→vers-le-bas et 1 →vers-le-haut). Si un paquet est reçu d'un nœud fils avec un bit marqué 0, le nœud parent en déduit l'existence d'une boucle (inconsistance) [14]. Il détruit le paquet et prend les mesures nécessaires pour éliminer la boucle. Ce bit de sens de progression du paquet devrait être utilisé pour toutes les données transitant par un réseau RPL.

Deux mécanismes de réparation sont prévus :

- la réparation locale:

  Dès qu'une boucle est détectée, le nœud concerné déclenche une réparation locale. Celle-ci fonctionne suivant le principe d'empoisonnement de la route. Le nœud se détache de son DODAG et annonce un rang de valeur infinie, ayant pour conséquence de ne pas le valider dans la liste des parents potentiels de tous ses fils. Après s'être détaché du DODAG [14], le nœud s'y attache à nouveau dès qu'il a trouvé un parent alternatif. Il recommence alors à annoncer un rang de valeur non définie déterminée à partir de celle de son nouveau parent.

- la réparation globale:

  La réparation globale impacte l'ensemble du DODAG. Elle est toujours déclenchée par la racine RPL et implique une reconstruction complète du graphe RPL. Une implémentation simple de celle-ci consiste pour la racine [14], à annoncer dans les DIO un numéro de version supérieur à la valeur courante.

c. Les fonctions objectives

La fonction objective permet de spécifier comment les métriques de routage sont transformées en rang. Elle est également responsable de la définition de la manière dont le nœud sélectionne le meilleur parent de la liste de ses parents potentiels. L'IETF n'a défini que deux fonctions d'objectifs pour RPL :

- OF0 (*Objective Function 0*):
  Elle implémente le nombre de sauts comme métrique, tous les liens participent avec le même poids pour la sélection du chemin.

- MRHOF (*Minimum Rank Hysteresis Objective Function*) :
  Elle est implémentée pour fonctionner avec les métriques additives [14]. Dans sa définition, l'IETF utilise l'ETX (nombre de transmissions attendu) comme métrique à optimiser, mais toute autre métrique additive (nombre de sauts, délai) pourrait être également utilisée. MRHOF se sert d'une hystérésis pour limiter les variations liées à l'emploi d'une métrique dynamique. Dans les documents d'accompagnement du standard RFC 6551 [14], plusieurs métriques sont prévues pour fonctionner avec le protocole, cependant la liberté est donnée au concepteur de définir comment les utiliser. Aucune recommandation n'est faite sur la façon de les combiner afin de prendre en considération la qualité de service (QoS) ou de les utiliser selon les spécificités liées à l'application et rendre les choses plus flexibles pour satisfaire les besoins des administrateurs.

5. Les Métriques utilisés pour choisir les chemins

De nombreuses métriques de routage ont été proposées pour les WSNs (*Wireless Sensor Networks*), l'économie d'énergie est l'un des buts principaux dans ce type de réseau notamment les LLNs (*Low Power and Lossy Networks*).

a. Le nombre de sauts

Cette métrique on la trouve dans pas mal de protocoles, elle est beaucoup utilisée pour des raisons de simplicité, le protocole de routage recherche le chemin le plus court en nombre de nœuds traversés. Les études expérimentales ont démontré que l'utilisation du nombre de sauts comme métrique de routage dans un WSN [14] peut résulter en de mauvaises performances. En effet, cette métrique a tendance à privilégier des liens entre nœuds éloignés (qui sont en général moins faibles) aux liens courts (beaucoup plus robustes) [14]. L'utilisation de liens de grande portée peut résulter en une consommation d'énergie plus importante que la concaténation de plusieurs liens courts.

b. L'ETX Expected transmission count

L'ETX est le nombre moyen de transmissions et retransmissions nécessaire pour qu'un paquet soit envoyé sur la liaison et soit correctement reçu par le destinataire. Afin de satisfaire les contraintes des LLNs, cette métrique est vraiment optimale, car non seulement elle améliore le taux de livraison global des paquets (en

sélectionnant les chemins les plus fiables), elle participe également à réduire l'énergie totale consommée (car évitant les liens nécessitant davantage de retransmissions).

Si Ps→d représente le taux de transmission avec succès des paquets de s vers d, alors l'ETX sur un saut est défini mathématiquement par la relation1.1 suivante [14]:

$$ETX = \frac{1}{P_{s\to d} \times P_{d\to s}} \quad \text{.... (1.1)}$$

c. RSSI (Received Signal Strength Indicator)

RSSI est une métrique de routage IPv6 pour RPL elle a été proposée sur pour améliorer les performances de transmission dans 6LoWPAN, le RSSI exploite les informations de la couche physique pour sélectionner les liens qui ont une meilleure qualité de signal reçu.

Toutefois, cette métrique n'est pas très précise et présente des zones pour lesquelles le taux de réception de paquets à la destination pour des valeurs similaires de RSSI varie énormément, Pour cette raison, la métrique LQI (*Link Quality Indicator*) [14] est plus souvent utilisée, en particulier avec des radios de type CC2420 où elle est beaucoup plus précise que le RSSI et permet d'obtenir un meilleur taux de réception de paquet.

6. Les défis du RPL

Il existe plusieurs problèmes qui doivent être traités avec soin pour une utilisation généralisée des applications LLN en temps réel. Certains des principaux moteurs de l'amélioration de la RPL sont : l'efficacité énergétique et la mobilité.

a. L'efficacité énergétique

L'un des problèmes les plus importants auxquels sont confrontés les LLNs est l'énergie limitée. Ce problème a été abordé par la minuterie de maintien qui minimise le nombre de messages de contrôle inutiles. Cependant, il est prouvé que la minuterie a ses propres inconvénients face aux environnements dynamiques, ce qui cause une perte d'énergie élevée due à l'échec de la livraison des paquets [15].

b. La mobilité

On cite trois problèmes principaux de performances avec RPL [19] :

- Le manque d'identification des nœuds mobiles : RPL ne fait pas la différence entre les nœuds mobiles et les nœuds non mobiles.

- les métriques proposées à l'IETF (comme MRHOF ou OF0) ne détaillent pas comment optimiser le routage en présence de nœuds mobiles,
- Une conception inhérente aux réseaux statiques: RPL intègre le mécanisme Trickle pour réduire le trafic de contrôle, Le problème général avec Trickle est que, en cas de stabilité temporaire dans une zone du réseau, le débit des messages de contrôle local peut diminuer à un point tel que la découverte des changements de topologie (dus à la mobilité) pourrait être plus lente que souhaité, Adaptabilité limitée, locale : lorsqu'un nœud détecte des changements (qui peuvent être plus tardifs que souhaité), RPL et Trickle réinitialisent les minuteries localement pour augmenter le temps de réponse jusqu'à ce que la topologie soit à nouveau stable. Cependant, le nœud qui détecte le changement de topologie n'est pas nécessairement le ou les nœuds qui devraient agir pour récupérer des modifications, et par conséquent, il peut ne pas être suffisant de mettre à jour les temporisateurs des messages de contrôle pour résoudre la situation de manière efficace.

Conclusion

Dans ce chapitre, nous avons défini l'internet des objets, les réseaux LLNs et le protocole RPL. Ainsi, le fonctionnement du protocole RPL et le métriques utilisées pour choisir les bons chemins de transmissions.

.

Chapitre 2
SDN pour l'internet des objets

I. Introduction

L'Internet des objets (IoT) est une technologie émergente qui permet à un écosystème intelligent d'exploiter des technologies hétérogènes. En général, les appareils physiques équipés de Tag RFID (Radio-identification), de capteurs sans fil et de dispositifs de communication sans fil sont connectés à Internet pour former un réseau IoT. Ces dispositifs sont spécifiquement déployés dans un contexte applicatif pour participer à la création d'un environnement intelligent allant par exemple à la communication Machine-to-Machine (M2M)[20], des réseaux véhiculaires aux réseaux de capteurs sans fil, en passant par les systèmes embarqués.

Selon le rapport de Cisco sur la croissance de l'IoT [22], le nombre d'appareils connectés à internet est passé de 6,4 milliards d'appareils en 2015 à 50 milliards en 2020.

Ces appareils connectés produisent une énorme quantité de données, de 6,2 exaoctets en 2015 a 30,6 exaoctets en 2020. Cette augmentation estimée de 781% dans les appareils connectés et 478% d'augmentation dans la génération de données en 2020, ce qui pousse à une anticipation d'une solution intelligente de contrôle et de gestion de réseau.

De nombreuses solutions ont été proposées pour résoudre les problèmes existants dans le paradigme de l'IoT. [20]Cependant, le réseau traditionnel n'est pas capable de gérer un nombre aussi énorme d'appareils connectés et d'énormes manipulations de données. Le réseau SDN (Software Defined Network) est considéré comme une technologie de réseau révolutionnaire qui prend en charge un réseau hétérogène avec une évolution et un dynamisme rapides à l'aide de plans programmables (Control Plane et Data Plane expliqués dans le chapitre précédent).

L'IoT et les SDN sont deux technologies distinctes. IoT principalement se compose de dispositifs réseaux (capteurs, microcontrôleurs) attribuant une communication différente, tandis que le SDN est associé au routage réseau et agit comme un orchestrateur [20] pour la gestion au niveau du réseau.

Par conséquent, l'IoT peut tirer parti des avantages du plan de contrôle SDN car le SDN promet d'établir une architecture réseau classique avec de nouvelles demandes de services, L'intégration SDN et IoT [20] peut répondre aux attentes de contrôle et de gestion dans divers scénarios.

Dans ce chapitre nous allons présenter les différents détails architecturaux de l'IoT compatible avec le model SDN et les différents protocoles utilisés ainsi que le fonctionnement de L'IoT dans un environnement SDN et au final nous allons voir les défis du SDN dans l'internet des objets.

II. Architecture du SDN

Contrairement à l'architecture classique, cette approche découpe le réseau en trois plans principaux : **le Control Plane, le Data Plane et l'Application Plane**.

Control Plane (plan de contrôle)

C'est le plan qui définit comment un équipement expédie le trafic dans le réseau. Le contrôleur est le composant principal responsable de l'établissement des tables de flux et du traitement des données, les politiques ainsi que l'abstraction de la complexité du réseau [16] et la collecte d'informations sur le réseau en communiquant avec la couche de dessous Data Plane.

Data Plane (plan de données)

C'est la couche d'architecture réseau qui gère physiquement le trafic en fonction des configurations qui lui ont été attribuées par le plan de contrôle, elle comporte tous les équipements physiques dont les switches, routeurs, points d'accès etc.

Application Plane (plan d'application)

C'est la couche qui contient les applications qui permettent d'exploiter toutes sortes d'avantages qu'apporte l'architecture, en introduisant de nouvelles fonctionnalités réseaux différentes.

Southbound Interface (Interface Sud)

Se sont les interfaces qui permettent le processus de communication entre le contrôleur et les switchs/routeurs et autres éléments de la couche de données, c'est grâce à cette interface, que le contrôleur injecte les différente politique aux équipements, et récupère les informations permettant aux applications de construire une vue globale du réseau (Goransson et al, 2016).Plusieurs protocoles adaptés pour d'interface sud, ils ont été développés pour interagir entre les deux couches Control Plane et Data Plane comme: BGP (*BorderGatewayProtocol*) flow-spec, Interface to Routing System (*I2RS*), NETwork CONFiguration protocol (*NETCONF*).

Northbound Interface (Interface Nord)

Contrairement aux interfaces vers le sud, les interfaces vers le nord permettent la communication entre les composants de niveau supérieur. Alors que les réseaux traditionnels utilisent un pare-feu ou un équilibreur de charge pour contrôler le comportement du plan de données, le SDN installe des applications qui utilisent le contrôleur et ces applications communiquent avec le contrôleur via son interface en direction nord. L'API Northbound permet aux opérateurs de réseau d'innover ou de personnaliser plus facilement les contrôles réseau et le traitement de cette tâche ne nécessite pas l'aide de l'expertise.

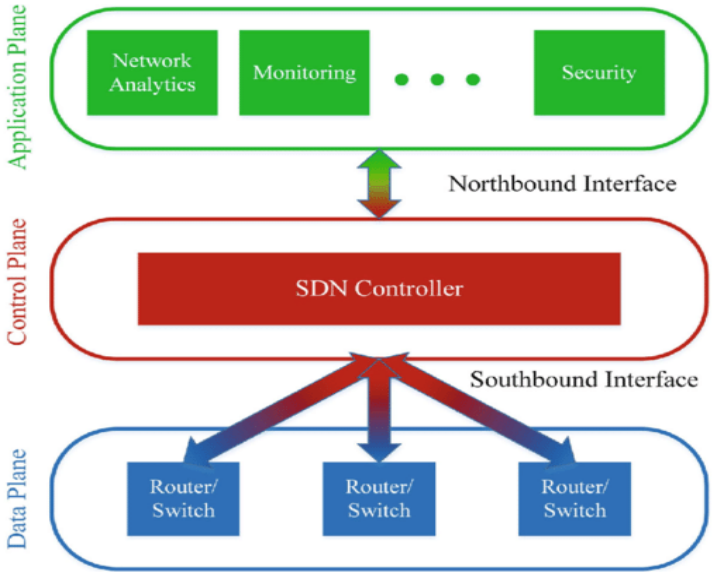

Figure 2.1- les interfaces du SDN[16]

III. Domaines et environnements de la solution SDN

Le SDN permet la personnalisation ainsi que le déploiement de nouveaux services réseau et stratégies dans divers environnements comme dans les:

- Data Centers : Des abstractions ont été proposées pour la consommation d'énergie dans les Data Centers, exploitent des réseaux à grande échelle, ce qui place l'application des politiques et la gestion du trafic à un niveau critique. Ces réseaux à grande échelle [18] sont soumis à divers défis car ils sont souvent complexes.

- Réseaux d'entreprise: Dans un réseau d'entreprise où la connexion de plusieurs appareils temporaires au réseau est courante. Cela pose un défi pour maintenir la sécurité et la gestion du réseau. Pour que les réseaux dans un environnement d'entreprise fonctionnent correctement, le modèle SDN peut être adopté pour faire appliquer les différents politiques et services.

- Domicile et petite entreprise: Bien que le SDN soit destiné aux réseaux à grande échelle, un certain nombre de recherches ont été faites concernant son utilité dans les petits réseaux tels que la maison ou les petites entreprises. Les réseaux à ce niveau utilisent des équipements à faible coût, d'où la nécessité d'une sécurité plus stricte et d'une gestion efficace du réseau. De plus, il n'est pas pratique d'avoir un réseau dédié opérateur dans chaque maison et bureau.

IV. Les réseaux traditionnels et SDN

Le tableau suivant représente la différence entre les réseaux traditionnels et les réseaux programmés par SDN :

Réseau traditionnel	SDN
➢ Le plan de contrôle et le plan des données résident sur le même dispositif. ➢ Un dispositif avec un plan de contrôle local doit être configuré manuellement et séparément. ➢ Aucun dispositif n'a la visibilité sur l'ensemble de réseau. ➢ Les nouveaux protocoles de routage ne peuvent pas être implémentés facilement. ➢ Intégration difficile des dispositifs de marques différentes pour fonctionner dans le même réseau.	➢ Un plan de contrôle commun est implémenté sur un dispositif de contrôle à distance (controller) pour tous les dispositifs du réseau. ➢ Les dispositifs deviennent simples car toutes prises des décisions sont effectuées par le contrôleur. ➢ On peut exécuter plusieurs systèmes d'exploitation sur des périphériques qui ne sont pas spécifiques à l'application.

Tableau2.1- Comparaison entre les réseaux traditionnels et SDN

V. Les avantages du SDN

Le SDN offre un large éventail d'avantage, nous citons :

- La reconfiguration du réseau: La programmabilité fournie par SDN permet la configuration des chemins de transmission dans le réseau, l'indépendance du protocole et le traitement personnalisé des flux individuels. Par exemple, cela permettrait de configurer des réseaux plus stables pour offrir de meilleures performances [21].

- Connaissance globale: Les réseaux sans fils contraints utilisent généralement des protocoles distribués afin de réduire la charge globale sur le réseau et de minimiser l'incertitude inhérente. Bien que dans cette approche, grâce aux connaissances globales, il existe un certain nombre de domaines dans lesquels les architectures centralisées pourraient apporter des avantages aux réseaux sans fil de faible puissance, en particulier dans la gestion des interférences et de l'hétérogénéité dans les zones denses et non réseaux isolés [21].

- Moins de consommation d'énergie: Le paradigme SDN est pratique car grâce au découplage, les nœuds de transfert (commutateur) sont libérés de la plupart des fonctionnalités de calcul. La plupart des fonctions consommatrices d'énergie résident désormais dans le contrôleur, qui dispose de suffisamment de ressources en énergie. Cela permet d'économiser une quantité d'énergie considérable et pourrait potentiellement prolonger la durée de vie du réseau [26].

- Routage, mobilité et localisation : La mobilité et la localisation sont essentielles pour un meilleur routage dans les réseaux de sans fil. Selon la nature du déploiement. Le SDN simplifie cela en gérant la mobilité à partir du contrôleur central, c'est-à-dire que les décisions et les politiques de routage sont gérées au niveau du contrôleur. Des algorithmes de localisation peuvent également être mis en œuvre au niveau du contrôleur ou au niveau du plan d'application au lieu des nœuds de capteurs à ressources limitées. Cela facilitera la découverte de la topologie du réseau et par la suite une meilleure prise de décision. [26]

- La sécurité : La centralisation de la gestion de la sécurité simplifie la configuration des mécanismes de sécurité. La vue globale du réseau permet les contre-attaques rapides au cas d'attaques. L'importance du SDN est que le nœud capteur devient un élément de vidage qui ne comprend que les messages du contrôleur, ce qui le rend difficile d'être utilisé comme conduit de malice.

VI. SDN pour les appareils IoT basés sur des capteurs sans fil

Les IoT intégrés au SDN sont réalisés sur une plateforme éducative et industrielle et de nombreuses études sont générées pour tirer partie de la programmabilité du SDN dans un réseau IoT. Dans cette section, nous présentons une vue d'ensemble sur les travaux faits dans le contexte SDN pour les appareils IoT.

- **SoftRAN (*soft Radio Access Network*):** Ce protocole est destiné aux réseaux cellulaires, il utilise le concept SDN dans le réseau 4G LTE. Soft RAN fournit une architecture pour la gestion coordonnée des ressources radio via son plan de contrôle logiquement centralisé. Avec une vue globale au réseau, une grande station de base gère les interférences, la charge, les contraintes de qualité de service afin de fournir une connectivité sans fil étendue aux clients mobile. Mais , il n'existe pas de solution concrète et le plan de contrôle centralisé et l'interaction entre le réseau central et le RAN ne sont pas définis [32].
- **SoftCell (*Soft Cellular*) :** il intègre le SDN dans le réseau central cellulaire et fournit des politiques à granularité fine pour le réseau LTE. Dans l'architecture SoftCell, la classification du trafic est effectuée sur les commutateurs d'accès au lieu de la passerelle. Chaque switch a un agent local SD-RAN qui contrôle la classification des paquets dans les commutateurs d'accès, ce qui offre un routage efficace [20].
- **SoftAir:** il intègre le SDN dans le réseau 5G en exploitant la virtualisation pour un réseau résilient. Il fournit un équilibrage de charge sensible à la mobilité et une allocation efficace des ressources grâce à la virtualisation. Toutes les politiques de gestion sont définies au niveau de plan de contrôle central qui fournit une garantie de qualité de service.
- **CellSDN:** il est dédié aux réseaux cellulaires. Les politiques basées sur les attributs sont formulées pour un utilisateur individuel dans le réseau LTE et obtiennent un contrôle sur le réseau. Les agents locaux sur chaque switch effectuent une inspection approfondie des paquets et réduisent la charge excessive sur le contrôleur.

- **Hybrid SDN-SDR :** pour le réseau cellulaire 5G. l'architecture est une combinaison intercouche de SDN et SDR pour exploiter le spectre de fréquence et les informations de liaison dans le réseau 5G.
- **SDN-WISE :** dans le cadre de la gestion des réseaux de capteurs sans fil (*WSN*), on a proposé le SDN-WISE dans lequel le SDN prend en charge le cycle d'utilisation et l'agrégation de données et fournit une solution à état complet. Son architecture est composée par le contrôleur qui est centralisé, la table de flux préinstallée. Il est très performant pour les environnements mobiles.
- **SD-WSN (*Software Defined – Wireless Sensor Network*):** les composants architecturaux de cette approche consistent en une station de base et plusieurs nœuds de capteurs. Le contrôleur qui fonctionne sur la station de base pris les décisions de routage et remplit les tableaux de flux des nœuds.
- **Integrated WSDN :** chaque nœud a son propre contrôleur local qui à son tour interagit avec le contrôleur central. Il offre une flexibilité lors de l'utilisation d'un appareil de base et réduit les couts. Il manque une évaluation de comportement et la performance de WSN.
- **SOF (*Sensor OpenFlow*) :** dans ce protocole, on a proposé le concept de reprogrammation et de re-tache dans WSN. La couche de contrôle qui est constituée de deux modules : « reconfiguration de capteur » et « contrôle de stratégie de requête » effectue un transfert basé sur le flux dans le plan de données qui est constitué de nœuds de capteur. l'idée de ce protocole reste théorique, elle n'est pas prouvée expérimentalement.

Tableau 2.2 – Comparatif des travaux connexes. [20]

Protocole	architecture	Mode de découplage	Scalabilité	Mobilité	Avantage	Inconvénient
SoftCell (cellular)	Routage MPLS	Contrôleur logiquement centralisé, agent local SD-RAN	Haute	-	Déchargement dynamique du trafic, routage efficace.	Politiques de service faibles.
SDN-WISE	Contrôleur centralisé avec nœud de capteur muet avec une table de flux qui est préinstallé.	Contrôleur centralisé	Moyenne	performant	State-full : l'échange d'informations réduit. Mobilité, reconfiguration.	En profondeur architectural les détails sont manquant et manque de sécurité
SoftRAN (cellular)	La gestion des ressources, trafic hors chargement	Contrôleur centralisé et agent local formant une grande station de base.	Faible	Performant	Gestion des ressources radio, support à la mobilité, déchargement du trafic, délai réduit.	Pas de solution concrète, la vitalisation n'est pas claire.
WSN-SDN	Cluster WSN avec contrôleur centralisé surveillé et contrôlé	Centralized master controller.	Faible	-	Sélection de chemin optimal, bonne stratégie de routage.	La mise en œuvre du contrôleur central n'est pas claire, aucune preuve de validation.

Protocole	Architecture	Mode de découplage	Scalabilité	Mobilité	Avantage	Inconvénient
SD-WSN	FPGA	Microcontrôleur	Faible	-	Reconfiguration programmable de réseau.	Matériel limité et périphérique dépendance.
Integrate WSDN	Contrôleur local dans chaque nœud de capteur qui interagit avec un contrôleur centralisé.	Contrôleur centralisé + contrôleur local	Faible	-	Utilisation flexible d'un appareil de base, réduire les couts.	Evaluation manquante pour le comportement et la performance du WSN.
cellSDN (cellular)	Etiquetage du trafic MPLS ou tag VLAN.	Contrôleur centralisé, agent de contrôle local BS.	Faible.	-	Performant.	Gestion transparente de la mobilité et contrôle du grain fin grâce à un agent local.
SOF	INPP i data plane.	Contrôleur centralisé et data plane distribué.	Faible	-	Gestion de la compatibilité avec les pairs, classification des adresses.	Idée théorique et pas expérimentalement prouvé.
Hybrid SDN-SDR (cellular)	Gestion du spectre.	Contrôleur centralisé	Faible	-	Economie d'énergie et optimisation	Contrôleur multicouche, maque de sécurité.
SoftAir (cellular)	Trafic distribué classification, Network managing.	SD-BS, SD-Switch, BS-clustering.	Haute	Performant	Plateforme flexible pour une architecture entièrement et partiellement centralisé.	Problème de sécurité.

VII. Les défis de SDN dans l'IoT

Les réseaux de capteurs IoT se composent généralement d'appareils contraints dans un réseau à faible consommation et avec perte (*LLN*) et sont limités en termes de fiabilité, de débit et d'énergie. La mise en œuvre d'une architecture SDN centralisée dans cet environnement est donc confrontée à des défis considérables.

Le SDN a, par nature, une architecture avec un surcoût associé élevé : à la fois en termes de trafic de contrôle centralisé, mais aussi de recherche de flux. [21]

- **Restrictions matérielles de l'appareil:** Les réseaux sans fil à faible consommation sont constitués d'appareils limités avec des capacités d'énergie, de mémoire et de traitement limitées. Ces restrictions permettent aux appareils de fonctionner pendant des mois, voire des années, avec peu d'énergie. Alors, des concessions doivent être faites à toutes les couches de la pile réseau. Cela est particulièrement limitatif pour le SDN, qui utilise traditionnellement des appareils capables de traiter des milliers de flux par seconde et de trier des tables pouvant parfois contenir des centaines de milliers d'entrées. Pourtant, les appareils IEEE 802.15.4 n'ont souvent que quelques Ko de mémoire, et une activité radio excessive épuisera rapidement l'alimentation en énergie d'un nœud.

- **Liens non fiables:** Le support avec perte présent dans les réseaux sans fil à faible puissance signifie qu'ils peuvent être sujets à un manque de fiabilité. Il s'agit d'une conséquence directe des exigences matérielles de faible consommation, qui obligent à faire des concessions au niveau des couches physique et MAC. Mais pour les applications SDN il faut des modèles de réseau à jour.

- **La fragmentation:** IEEE 802.15.4 a une unité de transmission maximale (*MTU*) de 127B. Après l'en-tête de couche liaison, la norme 6LoWPAN [7] introduit des capacités IP mais réduit davantage l'espace restant dans un seul paquet non fragmenté. Une implémentation IPv6 6LoWPAN complète avec adressage 64 bits permet seulement 53B de données d'application. Pour éviter la fragmentation, les messages de contrôle SDN doivent tenir la longueur allouée.

- **Interférence:** La nature de faible puissance des transmissions signifie que les réseaux IEEE 802.15.4 peuvent être sensibles aux interférences provenant de communications de plus haute puissance à proximité fonctionnant à la même fréquence. Cela peut potentiellement affecter des branches entières du réseau et entraver la livraison de messages depuis/vers les capteurs et les actionneurs. Dans une architecture SDN avec contrôle centralisé, cela empêche les nœuds d'interroger ou de recevoir des instructions du contrôleur.

- **Topologie de maillage multi-sauts:** Les protocoles de routage distribués, tels que RPL, sont couramment utilisés pour maintenir localement la topologie tout en réduisant la surcharge de contrôle dans l'ensemble du réseau.

Comme les appareils à faible puissance réduisent la portée radio, un maillage multi-sauts permet aux réseaux d'être étendus sur une plus grande zone que si tous les nœuds communiquaient avec une seule station de base. Malheureusement, en introduisant plusieurs sauts, l'incertitude de la liaison est aggravée sur la distance du saut et peut augmenter le risque de perte de paquets en cours de route.

VIII. Le concept SDN pour les LLNs

Le SDN peut faciliter la transmission de données élevées, l'efficacité spectrale, l'allocation des ressources et la gestion du réseau pour les appareils IoT afin de répondre aux besoins croissants [20] des demandes des clients, plus précisément Le contrôleur SDN offre une programmabilité et une flexibilité gestion de l'état de transfert de flux dans le plan de données en ayant une vue globale du réseau. Les avantages du SDN ont conduit à un très grand nombre de recherches pour appliquer le concept au sein du standard 802.15.4 de l'IEEE [21] pour réseaux sans fil à faible consommation, qui comporte les réseaux Internet des objets (*IoT*) et les réseaux de capteurs WSNs.

En particulier, la reconfigurabilité conférée par le SDN permettrait aux réseaux sans fil de faible puissance de traiter le trafic des capteurs et de contrôler le trafic de manière permanente, offrant des garanties aux données critiques tout en optimisant le réseau pour des communications à faible consommation d'énergie ce qui est voulu pour être compatible avec les besoins des réseaux LLNs (*low power and lossy networks*).

Le SDN est désormais considéré comme une ouverture pour les réseaux sans fil de nouvelle génération, en particulier dans les réseaux de capteurs IoT de faible puissance (*LLNs*), qui fonctionnent généralement dans un environnement extrêmement contraint. Plus précisément, dans la norme IEEE 802.15.4, les limitations de puissance obligent à utiliser un réseau maillé multi-sauts afin de permettre au réseau d'atteindre au-delà de la portée de transmission radio. Des réseaux typiques peuvent inclure des dizaines à des centaines d'appareils (dans un seul maillage) avec plusieurs capteurs par appareil.

Cependant, plusieurs réseaux peuvent être connectés sur un réseau fédérateur (backbone) et des protocoles tels que 6LoWPAN (vu dans le chapitre précédent) permettent aux appareils d'être interopérables avec IPv6. La flexibilité et l'évolutivité offerte par le SDN offrent une opportunité supplémentaire [21] d'aller au-delà des notions traditionnelles d'IoT de faible puissance en tant que petit réseau, ainsi que le SDN peut aider à distribuer les flux et à allouer des ressources réseau en fonction des exigences de QoS. En utilisant l'architecture SDN pour virtualiser le réseau des fonctions telles que le routage, la sécurité et l'agrégation de données, les réseaux de capteurs IoT peuvent tirer parti d'un meilleur calcul ressources au niveau du contrôleur. En plus de permettre aux fonctions d'être initialisé en fonction des besoins de l'application [21]. Ce processus permet en outre d'associer des flux à des fonctions individuelles. SDN peut permettre au réseau de servir dynamiquement plusieurs applications, telles que la collecte de données et l'activation, avec des exigences de qualité de service variables.

IX. Conclusion

Dans ce chapitre, nous avons vu le SDN, son architecture et ses avantages par rapport aux réseaux traditionnels, pour pouvoir présenter notre solution dans le chapitre suivant.

Chapitre 3
Proposition d'une solution basée sur les SDNs

I. Introduction

Les réseaux définis par logiciel (*SDN*) sont considérés comme un moyen de réduire la complexité de la configuration et de la gestion du réseau [23]. De nouvelles solutions de contrôle et de gestion de réseau peuvent être facilement déployées sur des équipements existants aussi simplement que pour installer de nouveaux programmes sur une machine.

Dans les réseaux SDN, les opérations de gestion sont centralisées et physiquement séparées des opérations de transfert. En fait, les commutateurs (nœuds dans notre cas) SDN classent et retransmettent les paquets selon les règles dites de flux envoyées par le (s) contrôleur (s) SDN. Lorsqu'un commutateur n'a aucune information disponible pour classer un paquet entrant, il demande l'assistance d'un ou plusieurs contrôleurs qui devraient fournir une règle appropriée [23]. Les politiques appliquées par les contrôleurs, qui définissent finalement l'ensemble du comportement du réseau, peuvent être modifiées facilement et rapidement en installant un nouveau logiciel de contrôleur. Cela est interprété par la solution que nous proposons dans ce travail qui est intitulé: **SDN-WISE**.

II. L'architecture du protocole

SDN-WISE est basé sur les couches physiques et MAC IEEE 802.15.4. Les éléments de réseau peuvent être distingués en sinks et nœuds. Ces éléments sont équipés d'une interface réseau connectée à un réseau infrastructurel. Par conséquent, tous les paquets de contrôle doivent trouver leur chemin vers le pour quitter le WSN et atteindre le contrôleur.
L'architecture de protocole de SDN-WISE est illustrée dans la figure 3.1.

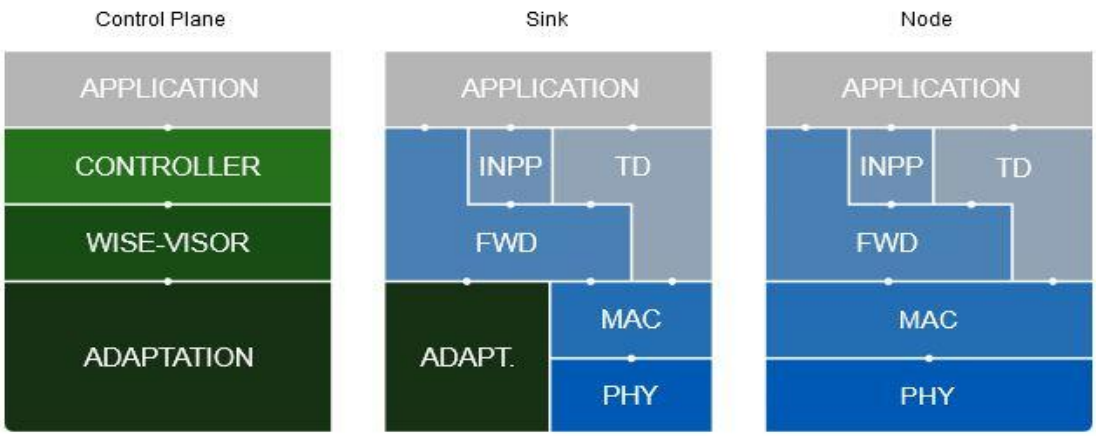

Figure 3.1- l'architecture de SDN-WISE[23]

Au-dessus de la couche MAC, la couche de transfert (*FWD*) gère les paquets entrants comme spécifié dans la WISE Flow Table. La couche FWD met à jour cette table en fonction des configurations envoyées par le plan de contrôle. La couche INPP (*In-Network Packet Processing*) s'exécute au-dessus de la couche Forwarding et est responsable des opérations telles que l'agrégation de données ou tout autre traitement en réseau. Si aucune entrée dans la table de flux WISE ne correspond au paquet actuel, une demande est envoyée au plan de contrôle. Afin de contacter le plan de contrôle, chaque nœud doit connaître son meilleur prochain saut vers le sink. Cette valeur est calculée de manière distribuée à l'aide de la couche Topology Discovery (*TD*) par le processus du beaconing.

Dans le plan de contrôle, les logiques du réseau sont dictées par un ou plusieurs contrôleur (s) et un WISE-Visor. Le WISE-Visor fait abstraction des ressources réseau afin que différents réseaux logiques, avec des stratégies de gestion différentes définies par différents contrôleurs, puissent s'exécuter sur le même ensemble de périphériques physiques. Entre le sink et le WISE-Visor se trouve la couche Adaptation qui est responsable du formatage des messages reçus du Sink de manière à ce qu'ils puissent être traités par le WISE-Visor et vice-versa[23].

III. Le protocole SDN-WISE

1. Découverte de topologie

La découverte de topologie (*TD*) est exécutée par tous les nœuds capteurs, elle est responsable de la génération de ces informations et de leur transmission au WISE-Visor. Le protocole TD conserve les informations sur le prochain saut vers le contrôleur et ses voisins actuels mis à jour. Dans ce but, tous les récepteurs du réseau SDN-WISE transmettent périodiquement et (presque) simultanément un paquet de découverte de topologie (paquet TD) sur le canal sans fil de diffusion. Un tel paquet contient l'identité du nœud qui l'a généré, un niveau de batterie et la distance actuelle du sink qui est initialement fixée à 0[24].

Un nœud de capteur **A** recevant un paquet TD du nœud de capteur **B** (notez que B peut être le sink) effectue les opérations suivantes:

1) Insère dans la liste de ses voisins actuels avec le RSSI actuel et le niveau de la batterie. Évidemment, si B est déjà présent dans la liste des voisins actuels, alors seules les valeurs RSSI et niveau de batterie sont mises à jour.

2) Contrôle s'il a récemment reçu un paquet TD avec une valeur inférieure de la distance actuelle du sink. Si ce n'est pas le cas, alors le nœud met à jour la valeur rapportée dans le paquet TD à la valeur actuelle plus un et définit son prochain saut vers le contrôleur égal à B.

3) Règle son niveau de batterie dans le champ correspondant du TD Packet.

4) Transmet le paquet TD mis à jour sur le canal de diffusion sans fil.

Périodiquement, chaque nœud de capteur génère un paquet contenant sa liste actuelle de voisins et l'envoie au WISE-Visor. Notez que la liste des voisins est périodiquement effacée. Les nœuds recevant des paquets dirigés vers le contrôleur les relient au nœud défini comme leur prochain saut vers les contrôleurs.

2. Packet building

Comme dans chaque protocole, un format de paquet est respecté, SDN-WISE a mis en œuvre un format qui contient des champs bien précis la figure en dessous présente le header du **WISE-Packet :**

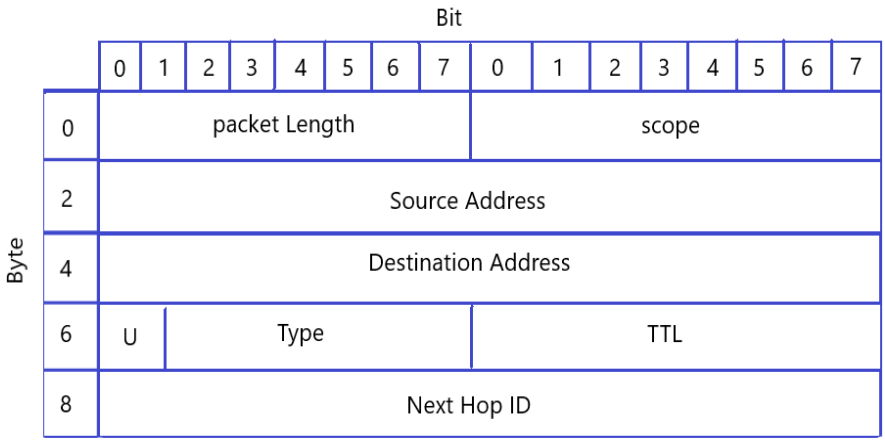

Figure 3.2- Architecture du paquet SDN-WISE

Les paquets SDN-WISE ont un en-tête fixe composé de 10 octets répartis dans les champs représentés dans le tableau 3.1 :

Packet Length	la longueur du paquet, y compris la charge utile (le cas échéant), en octets.
Scope	identifie un groupe de contrôleurs qui ont exprimé leur intérêt pour le contenu du paquet. La valeur du Scope est initialement définie sur 0 (par défaut) mais peut être modifiée par des entrées appropriées dans la table de flux WISE du nœud de capteur générant le paquet. Dans notre implémentation actuelle, les valeurs Scope ont une validité globale car la couche WISE-Visor garantit la cohérence à l'échelle du réseau.
Source and Destination Adresses	spécifie évidemment les adresses du nœud qui a généré le paquet et la destination prévue.
Flag U	marquer les paquets qui doivent être livrés au sink le plus proche.

Type of Packet	Pour distinguer entre différents types de messages en fait en plus des paquets de données, des paquets TD et des paquets contenant des informations de topologie locale, SDN-WISE utilise d'autres types de paquets pour la demande d'une nouvelle entrée aux contrôleurs, pour l' introduction d'une nouvelle entrée dans la table de flux (Flow Table) d'un nœud de capteur donné, pour ouvrir un chemin dans une séquence de nœuds de capteur, et pour désactiver l'interface sans fil d'un nœud de capteur pendant un certain intervalle de temps. Le type de paquet déterminera l'interprétation de la charge utile du paquet.
TTL (*Time To Leave*)	Le temps de vie qui est réduit d'un à chaque saut.
Next Hope ID	Ce champ doit être présent dans le tableau d'identifiants acceptés pour que le paquet soit traité ultérieurement par le nœud de capteur.

Tableau 3.1 : Champs du paquet SDN-WISE

3. La WISE FlowTable

Matching Rule					Matching Rule					Matching Rule					Action					Statistics	
Op.	Size	S	Addr.	Value	Op.	Size	S	Addr.	Value	Op.	Size	S	Addr.	Value	Type	M	S	Addr.	Value	TTL	Counter
=	2	0	2	B	>	2	0	10	x_{thr}	=	1	1	0	0	Modify	1	1	0	1	122	23
=	2	0	2	B	≤	2	0	10	x_{thr}	=	1	1	0	1	Modify	1	1	0	0	122	120
=	2	0	2	B	-	0	-	-	-	-	0	-	-	-	Forward	0	0	0	D	122	143
=	2	0	2	A	=	1	1	0	0	-	0	-	-	-	Drop	0	0	-	-	100	42
=	2	0	2	A	=	1	1	0	1	-	0	-	-	-	Forward	0	0	0	D	100	32

Figure 3.3 -La Table de flux [23]

Dans SDN WISE la structure de la flow table comporte trois champs principaux: Matching rule (Règles de correspondance), les actions et les statistiques. Les règles de correspondance spécifient jusqu'à trois conditions. Si ces conditions sont satisfaites, l'action correspondante est exécutée et les informations rapportées dans la section Statistiques sont mises à jour.

Chaque règle de correspondance consiste en un champ (S) qui spécifie si la condition concerne le paquet courant (S=0) ou l'état (S=1) ; les champs Offset et Taille spécifient respectivement le premier octet et la taille de la chaîne d'octets du paquet ou l'état à considérer, le champ Opérateur donne l'opérateur relationnel à vérifier par rapport à la Valeur

donnée dans la règle [24]. Par exemple, la deuxième règle de correspondance dans la première entrée dans la table de flux WISE donnée sur la figure est satisfaite si les 2 premiers octets (Taille = 2) après l'octet 10 (Offset = 10) du paquet actuel (S = 0) prennent une valeur qui est supérieur (Op = « > ») à xThr (Valeur = xThr). Si toutes les conditions spécifiées dans la section Règles de correspondance sont satisfaites (si Taille = 0, la règle de correspondance n'est pas prise en compte), l'action correspondante est exécutée.

L'action est spécifiée par cinq champs. Le Type spécifie le type d'action. Les valeurs possibles du champ Type peuvent être « Forward to (Transférer vers)», « Drop (Abandonner) », « Modify (Modifier) », « Send to (Envoyer vers) INPP », « Turn off radio (Désactiver la radio) ». Le flag M spécifie si l'entrée est exclusive (M=0) ou non (M=1). Dans le premier cas, si les conditions sont remplies, le nœud capteur exécute l'action puis arrête de parcourir la table de flux WISE. Dans le second cas, à la place, après avoir exécuté l'action, le nœud capteur continue de parcourir la table de flux WISE et exécute d'autres actions si les conditions correspondantes spécifiées dans la section Règles de correspondance sont satisfaites. La signification des deux autres champs (Offset et Value) dépend du type d'action. Par exemple, si l'action est « Transférer vers », ils doivent spécifier quel est l'ID du prochain saut (qui sera écrit dans le paquet), s'il s'agit de « Abandonner », ils donnent la probabilité de chute ainsi que l'ID du prochain saut au cas où le paquet n'est pas abandonné, s'il s'agit de « Modifier », ils spécifient l'Offset et la nouvelle valeur à écrire, s'il s'agit de « Envoyer à INPP », ils précisent le type de traitement qui doit être exécuté, s'il s'agit de « Éteindre la radio » ils spécifient après combien de temps la radio doit être rallumée [24].

IV. Avantages et bénéfices de l'utilisation de SDN-WISE

1. **Efficacité énergétique**

   Cela prend en compte la capacité du système à conserver l'énergie ou à permettre un fonctionnement avec une puissance limitée pendant de longues périodes, ce qui améliore la durée de vie du réseau, SDN-WISE supporte le duty cycle il permet aux nœuds non actifs d'éteindre leur interface radio afin de conserver un max d'énergie, ainsi qu'il limite la consommation d'énergie lors de l'envoi et la réception.

2. **Robustesse:**

   Il garantit que le système fonctionne comme prévu, quelles que soient les conditions environnementales ou les exigences de conception variables. SDN-WISE produit des performances souhaitables malgré les variations du réseau telles que les pannes de nœuds, les pannes de courant et les instabilités résultant de la mobilité des nœuds qui engendrent la perte des paquets. Une caractéristique importante d'un système de gestion robuste est la reconfiguration du réseau.

3. **Évolutivité :**

   Un système de gestion évolutif devrait fonctionner efficacement à n'importe quelle échelle de réseau. La gestion distribuée joue ici un rôle important tout en réduisant la surcharge de trafic qui pourrait autrement être entièrement dirigée vers un gestionnaire d'activité centralisé, grâce au WISE-VISOR (définit précédemment) qui s'occupe de la gestion du réseau, l'évolutivité ne cause pas un problème.

4. **Adaptabilité :**

SDN-WISE capable de fonctionner efficacement dans des conditions de réseau variables telles que les fluctuations d'énergie, les changements de topologie et la variation des tâches. La capacité de reconfigurer et de redéfinir les tâches joue également un rôle important pour répondre à ces critères.

5. **Statefulness :**

SDN-WISE est statefull un tampon de mémoire est réservé aux informations d'état. Les règles peuvent utiliser les informations d'état pour classer les paquets dans les flux Les actions peuvent modifier les informations d'état, afin de réduire le nombre d'interactions avec le Contrôleur si des politiques locales doivent être appliquées.

- Exemple sur la Statefulness (qualité de service): Un nœud encombré doit donner des priorités différentes aux différents flux Niveau de congestion stocké comme information d'état Différentes probabilités de chute (Drop) données aux différents flux dans la table WISE en cas de congestion [25,26].

V. Conclusion

Dans les prochains chapitres nous allons pouvoir implémenter SDN-WISE et tester les performances de ce dernier afin de critiquer ce système avec différents paramètres d'évaluation dans de différents environnements.

Chapitre 4
Evaluation des Performances

I. Introduction

Dans les chapitres précédents, nous avons expliqué la structure, la conception et l'implémentation du protocole SDN-WISE qui a été développé pour intégrer le model SDN dans l'IoT afin d'améliorer les performances du réseau. Dans ce chapitre, nous présentons les résultats obtenus lors de la simulation du protocole SDN-WISE en comparaison avec le protocole RPL sous le simulateur Cooja.

II. La Mobilité des Nœuds

Une partie de la simulation a été faite dans un environnement mobile, c'est à dire que l'ensemble des nœuds changent de position d'un moment à un autre ce qui va influencer toute la topologie et les différents paramètres de routage ce qui donne des résultats différents d'un environnement Immobile.

Dans la vie réelle, la mobilité représente un challenge dans les réseaux, comme on peut le voir dans les réseaux de type MANET et VANET ou n'importe quel réseau Ad-Hoc. Dans notre cas le modèle de mobilité que nous avons implémenté s'appelle le: **"Random Waypoint Model"**. Il appartient à la famille **"RANDOM - BASED MOBILITY MODELS".** Dans cette famille, les nœuds mobiles se déplacent de manière aléatoire et libre sans restrictions. Pour être plus précis, la destination, la vitesse et la direction sont toutes choisies au hasard et indépendamment des autres nœuds [27]. Ce type de modèle a été utilisé dans de nombreuses études de simulation. Le modèle **Random Waypoint** a été proposé pour la première fois par Johnson et Maltz [27]. Bientôt, il est devenu un modèle de mobilité « de référence » pour évaluer les protocoles de routage MANET, en raison de sa simplicité et de sa grande disponibilité.

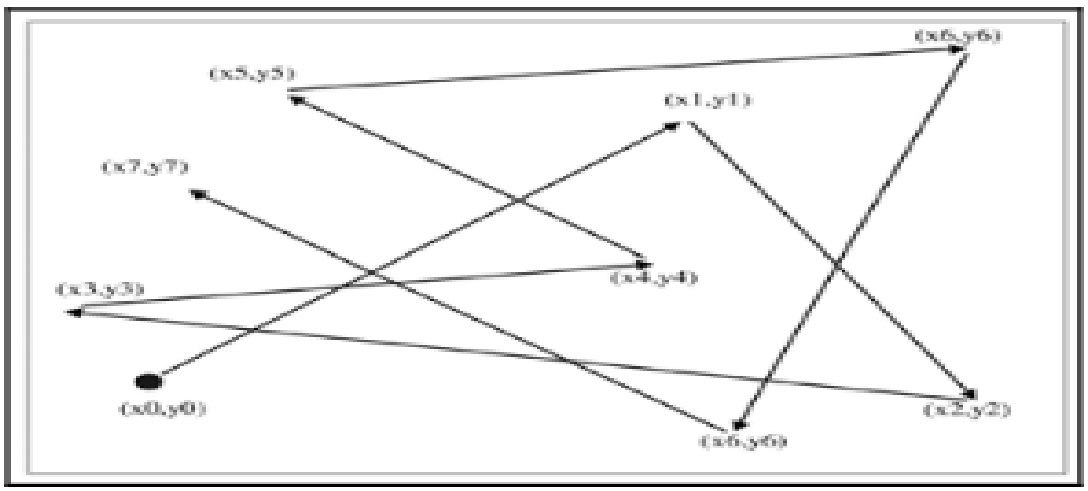

Figure 4.1- Exemple de mouvement de nœud dans le Random Waypoint Model [27].

III. Topologie et Configuration

Afin de tester le protocole SDN-WISE, plusieurs simulations ont été faites sur COOJA. Chacune comporte des paramètres différents, l'évaluation a été faite en se basant sur des métriques différentes dans deux environnements principaux : E. Mobile et E. non Mobile. Ces simulations testent les communications Point-to-Point (*P2P*) dans une zone qui contient des nœuds aléatoirement déployés, on constate deux types de nœuds: le Sink ou le nœud racine et le nœud ordinaire.

Dans SDN WISE Le nœud Sink est le premier nœud du réseau à recevoir les messages envoyés. Il vérifie dans sa table de flux WISE, il ne trouve pas de règle de correspondance, puis il demande au contrôleur. Le contrôleur fournit un chemin pour atteindre la destination et envoie les règles correspondantes. Les nœuds de ce chemin apprennent les règles et peuvent ainsi transmettre correctement le message.

Lorsqu'un nœud reçoit un message pour lui-même, il imprime la charge utile dans la fenêtre de sortie Mote-Output dans COOJA.

Dans RPL on a un réseau classique qui ne contient pas de contrôleur les nœuds envoient régulièrement des paquets au sink afin d'établir la connexion. L'ensemble des nœuds s'accorde pour former le DODAG grâce au processus du protocole RPL vu auparavant. Une fois le réseau converge, ils commencent à envoyer au sink des paquets dont le payload est : Hello.

Les Simulation ont été faites selon plusieurs charges de nœuds et d'autres paramètres détaillés dans le tableau 5.1 :

Paramètre	**Valeur**
Simulateur	COOJA
Contiki	Version 3.0
Plateforme	Skymote
Charge de nœud	10, 15, 20,30
Zone de transmission	50m
Zone d'interférence	100m
Taille de la grille	900m²
Success Ratio TX/RX	100% / 100%
Model Radio	UDGM Distance_loss
Mobility Type	Random Waypoint Model

Tableau 4.1 : Paramètres de la simulation

La simulation a été faite dans un environnement où la probabilité de la transmission et la réception est maximisée à 100%, les nœuds sont répartis dans une surface de 900 m² avec une zone de transmission de 50m et une zone d'interférence de 100m.

Le modèle radio **UDGM Distance_Loss [28]** est un modèle dans lequel la plage de transmission est considérée comme la meilleure région ou les nœuds peuvent recevoir les paquets avec succès et tous les nœuds situés derrière cette dernière ne reçoivent pas de paquets. Le taux de réussite de l'émission et de la réception peut être défini par TX/RX. Les paquets sont transmis avec la probabilité TX et reçus avec la probabilité RX.
Le type de plateforme de nœuds utilisé dans cette simulation est nommé **Skymote** [29] voici ses caractéristiques:

-Très faible consommation d'énergie.
-Compatible avec les périphériques qui utilisent le standard IEEE 802.15.4
-48KB de mémoire ROM
-Interopérable avec le système du Duty Cycle.

IV. Métriques évaluées

Deux principales métrique sont prise en considération durant la simulation elles sont évalués dans deux environnements différents (Mobile et Fixe):

1. **Packet Reception Rate (PRR):** C'est la relation entre le nombre de paquets reçu avec le nombre de paquets envoyés. Il est définit par l'équation suivante:

$$PRR = \frac{\sum_{i=1}^{i=N} Pr}{\sum_{i=1}^{i=N} PS} \quad \text{.... (5.1)}$$

-N : le nombre de paquets total envoyé.
-Pr : le nombre de paquets reçu.
-Ps: le nombre de paquets transmis.

2. **la consommation d'énergie durant la réception (*Erx*)**

C'est la moyenne de l'énergie de réception consommée par tous les nœuds durant la simulation. D'abord elle est calculé par l'équation 5.2 [30] puis elle est convertit à un pourcentage par rapport à l'énergie totale du nœud:

$$\textbf{\textit{Power (mW) = (rxend – rxstart) * 20mA * 3V / 4096}} \quad \text{..... (5.2)}$$

-rxstart: le temps du début de la réception.
-rxend: le temps de la fin de la réception.

IV. Evaluation des Performances

Durant notre simulation nous avons comparé le protocole SDN-WISE au protocole RPL pour voir s'il est mieux en termes de réception de paquets et aussi pour comparer la consommation d'énergie des deux durant la réception. Le nombre total de simulation effectué est d'environ 12 simulations dont un scénario ou les nœuds sont mobiles. Les figures 5.1 et 5.2 représentent la topologie des deux protocoles dans une des 12 simulations (15 nœuds dans celle-là) ainsi que le mote output des deux protocoles où le payload des paquets est affiché. Dans Cette évaluation de performance nous considérons que l'énergie est illimité et que la sécurité est bien maintenue.

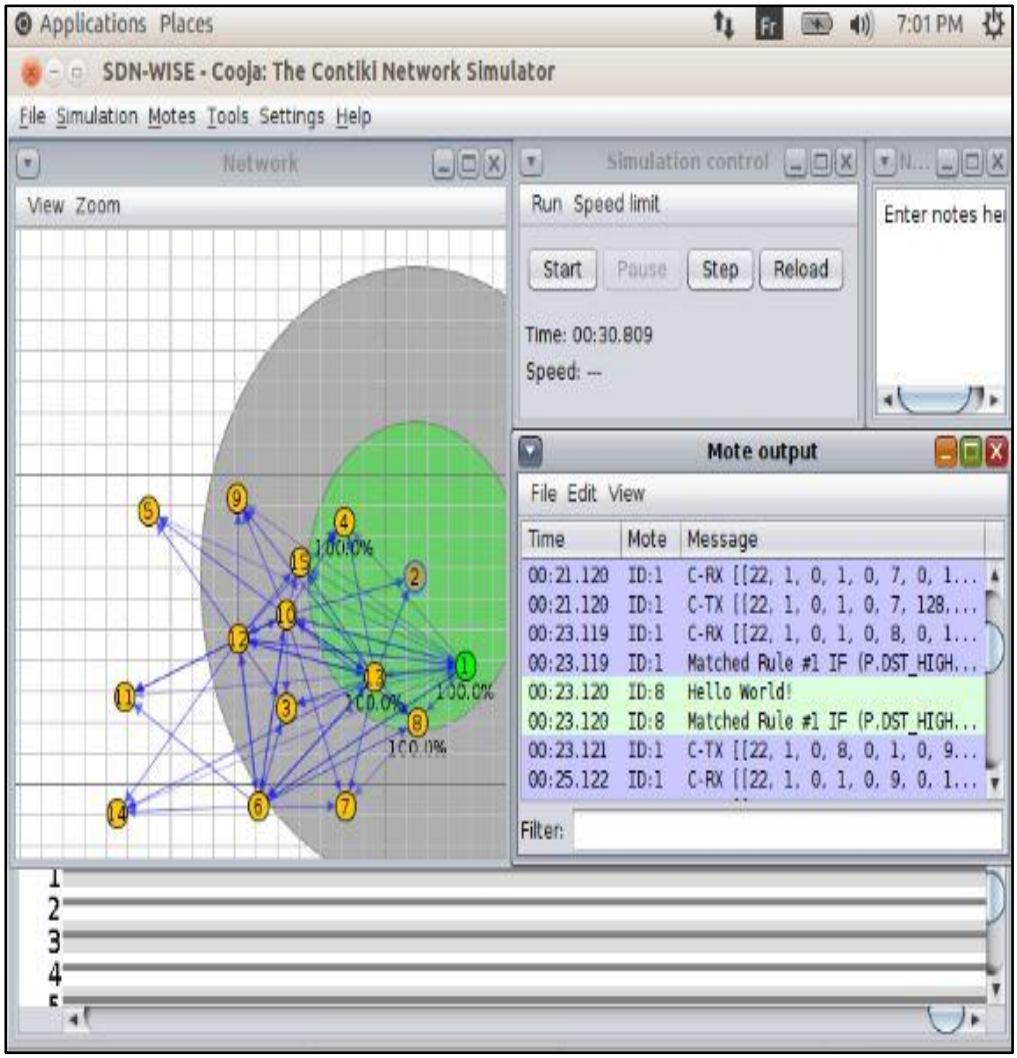

figure 4.2 Topologie SDN-WISE

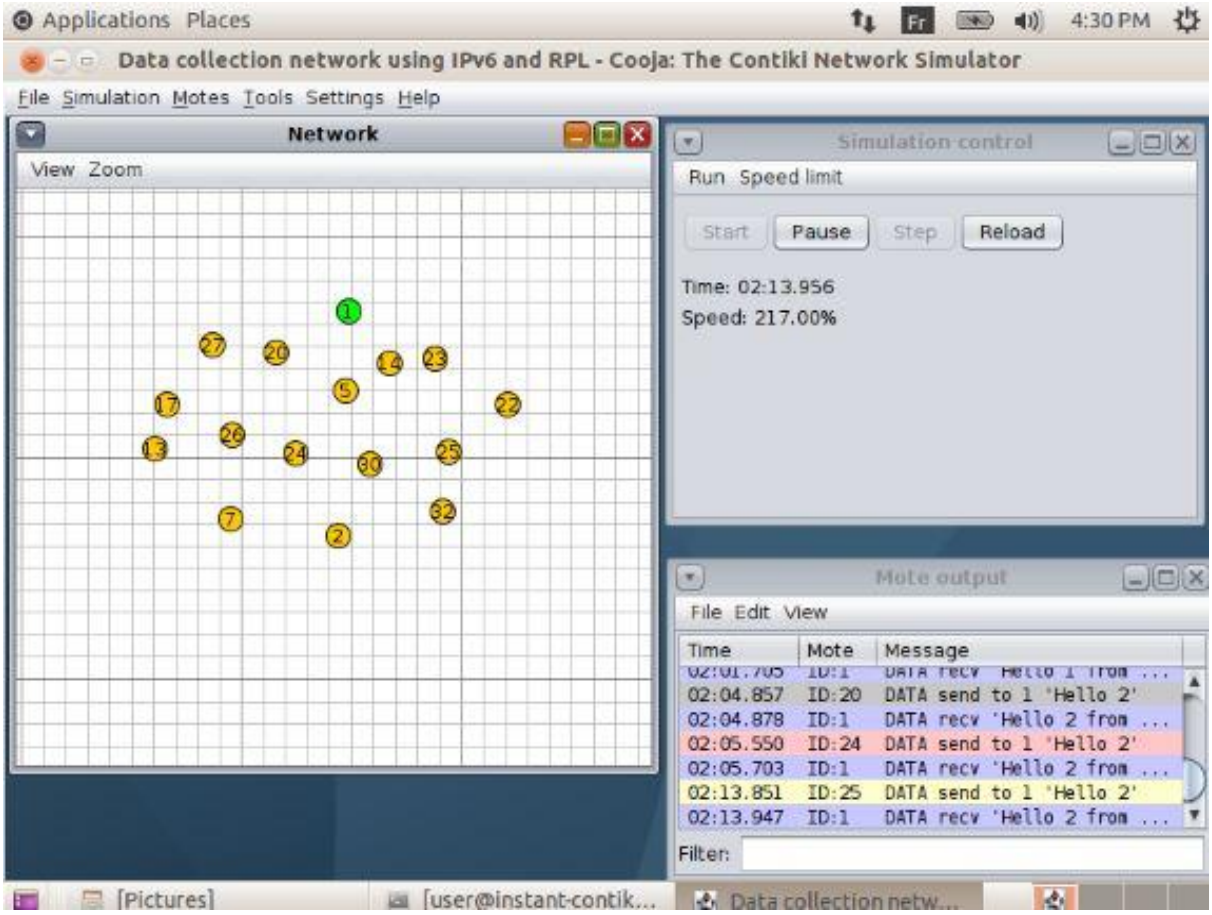

Figure 4.3 : Topologie RPL

1. **Les résultats de la simulation**

Les résultats de la simulation sont représentés graphiquement sur les figures suivantes dans un premier temps on teste SDN-WISE contre RPL pour la métrique PRR dans un environnement Fixe, en deuxième lieu on refait la même simulation mais cette fois ci dans un environnement Mobile (*Random Waypoint Model*) et en dernier lieu on simule dans un environnement fixe en évaluant la consommation d'énergie durant la réception.

A. Environnement statique

➢ PRR

Dans la figure 4.3 nous remarquons que RPL arrive à transmettre tous les paquets avec succès, la valeur du PRR est égale à 1 dans toutes les charges de nœuds, tandis que SDN-WISE a donné des valeurs entre 0,9 et 0,98. Cette petite perte de paquets est due aux collisions entre les paquets envoyés, par contre dans RPL la probabilité de collision est très faible ou parfois nulle, car ce protocole utilise la technique du Trickle Timer, grâce à cette méthode les nœuds n'envoient pas des paquets simultanément donc le risque d'engendrer une collision est très faible ce qui diminue la perte de paquet.

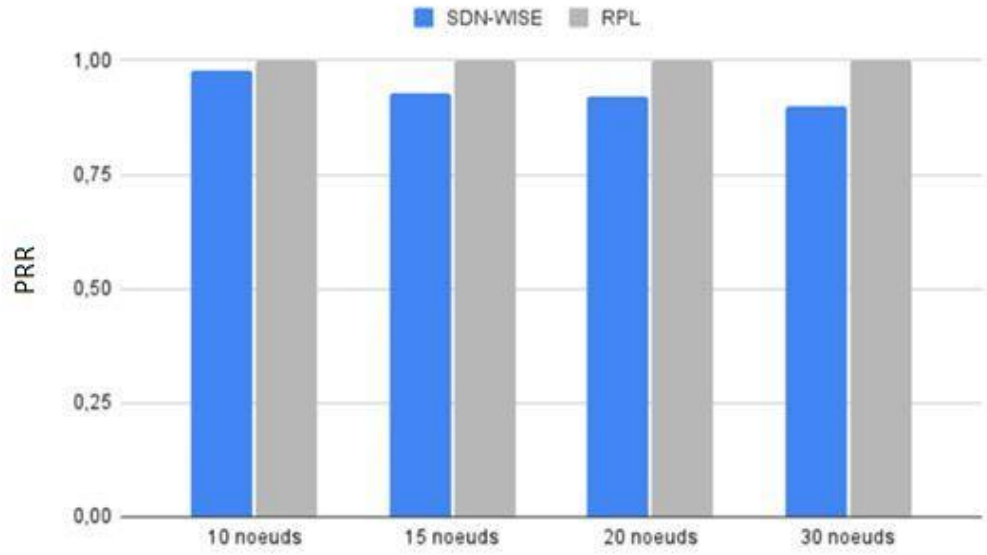

Figure 4.4 : PRR environnement Fixe

➢ **Consommation d'énergie lors de la réception**

D'après le graphe obtenue dans la figure 4.5 nous remarquons que SDN-WISE consomme moins d'énergie que RPL lors de la réception, respectivement les valeurs sont: [0,12 %.. 0,24%] et [0,15%.. 0,6%].

Cela peut être expliqué par le traitement lors de la réception, SDN-WISE reçoit moins de message de contrôle que RPL donc il traite moins de donné lors de la réception ainsi que la technologie du Duty Cycle intégré dans SDN-WISE permet de limiter la consommation d'énergie dans l'interface radio d'un nœud.

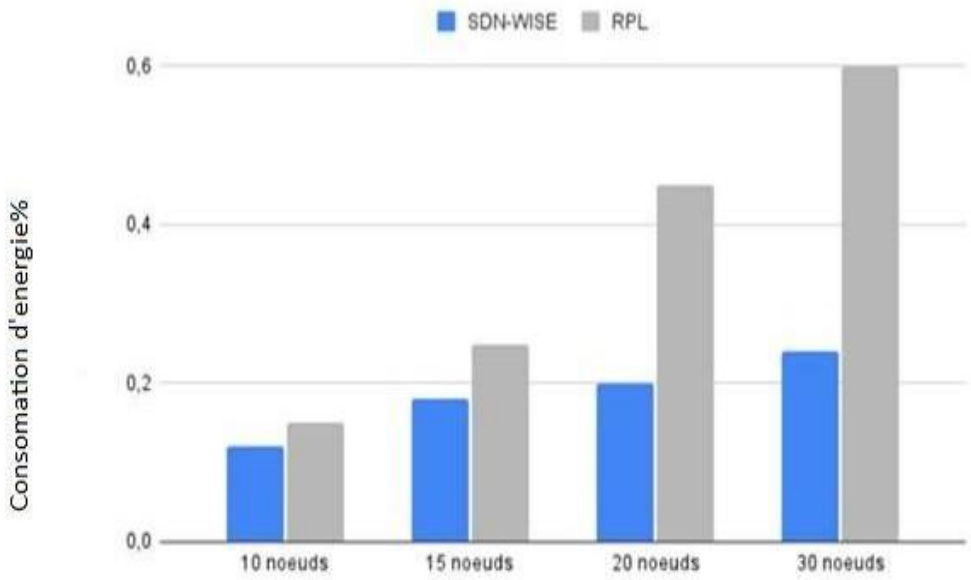

Figure 4.5 : Consommation d'énergie a la réception

B. Environnement mobile

- **PRR**

Dans l'environnement mobile nous avons obtenu une grande différence dans les valeurs en comparant avec l'environnement fixe. Nous remarquons dans la figure 4.6 que RPL a enregistré des valeurs comprises entre 0,27 et 0,58. Cela est due au processus de reconstruction de la topologie car dans un environnement mobile il est quasi impossible de maintenir un DODAG et la technique du Trickle Timer n'est plus valable dans ce cas la perte est très élevé à cause des collisions arrivés et aussi au mal acheminement des paquets dans le réseau.

Par contre SDN-WISE arrive à transmettre la majorité des paquets avec succès dans un environnement mobile, il enregistre de très bonnes valeurs comprises entre 0,6 et 0,9. Cette plage est un peu inférieure au résultat de SDN-WISE dans un environnement fixe mais ca reste bien meilleure que RPL, car le routage dans ce cas se fait à l'aide du contrôleur, le processus de reconstruction de la topologie est bien plus rapide dans SDN-Wise en le comparant avec RPL.

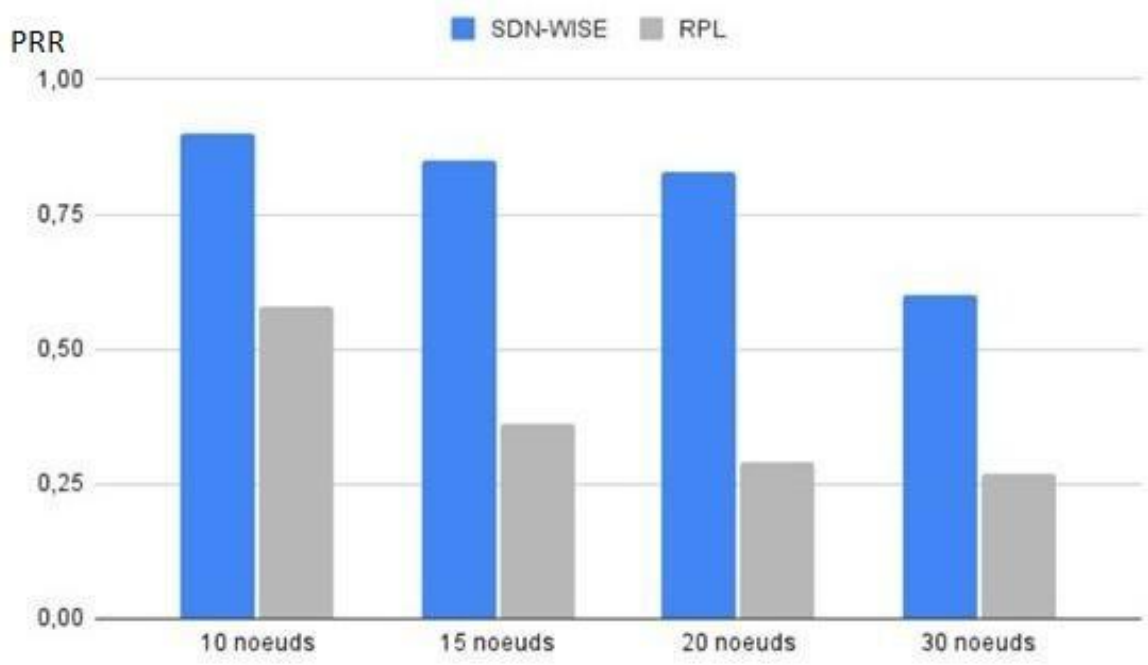

Figure 4.6 : PRR environnement mobile

➢ **Consommation d'énergie lors de la réception**

D'après le graphe obtenue dans la figure 4.7 nous remarquons que SDN-WISE consomme moins d'énergie que RPL lors de la réception, respectivement les valeurs sont: [0,15 %.. 0,5%] et [1%.. 3%].

Cela peut être expliqué par le traitement lors de la réception, SDN-WISE reçoit moins de message de contrôle que RPL donc il traite moins de donné lors de la réception ainsi que la technologie du Duty Cycle intégré dans SDN-WISE permet de limiter la consommation d'énergie dans l'interface radio d'un nœud.

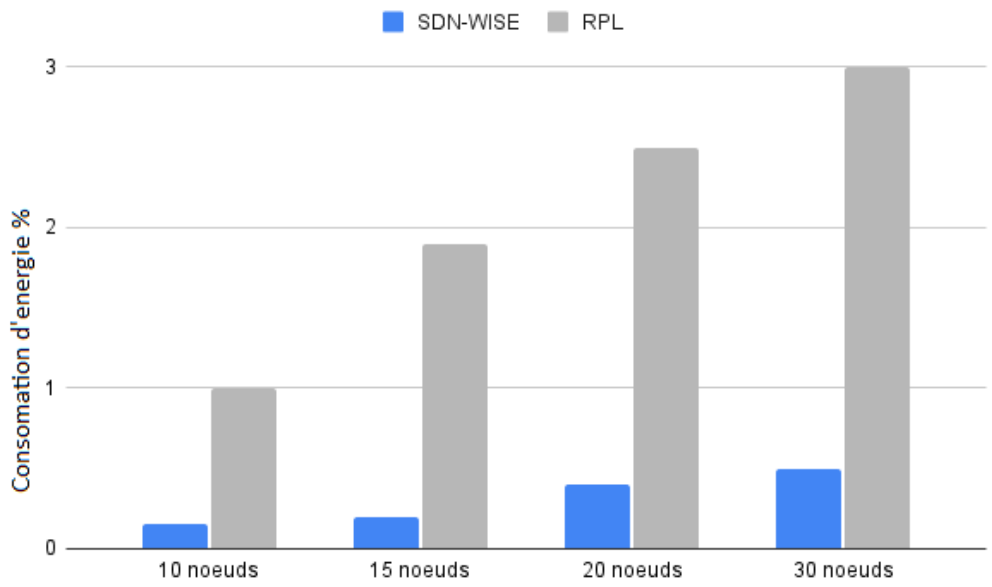

Figure 4.7 : Consommation d'énergie a la réception environnement mobile

V. conclusion

Dans ce chapitre nous avons fait une évaluation de performance du protocole SDN-WISE après de l'avoir implémenté sous COOJA. Nous avons fait également une comparaison analytique entre SDN-WISE et RPL les résultats ont déterminé que la solution qu'on a choisi est très favorable, surtout dans le cas mobile SDN-WISE est beaucoup plus performant que RPL en terme PRR, ainsi qu'il consomme moins d'énergie que RPL. Sauf dans le cas Fixe on a remarqué une légère différence, dans ce cas RPL dépasse un peut SDN-WISE mais ça reste qu'une petite variante.

Conclusion générale

Les réseaux LLN sont utilisés dans plusieurs domaines : villes intelligentes, grilles intelligentes, santé, agriculture, industrie 4.0, etc.
Si de tels réseaux offrent beaucoup d'avantages, il reste, par contre, plusieurs défis à résoudre pour ce type de réseaux et pour chaque cas d'utilisation on trouve un protocole qui plus adapté et plus adéquat pour le mettre en œuvre.

Dans le cadre de notre travail nous nous somme intéressés de plus près aux solutions de type SDN (*Software Defined* Network) qui sont adapté pour les réseaux LLN. En premier lieu, nous avons étudié plusieurs travaux et propositions qui traitent le domaine des SDNs dans leur sujet, nous avons également étudié le protocole RPL (*Routing for LLNs*) qui est utilisé dans ce type de réseaux et enfin nous avons comparé les deux protocoles selon plusieurs métriques.

Concernant la solution SDN, nous nous sommes intéressés à SDN-WISE qui est un choix très performant en termes de fiabilité de routage. Ce choix est justifié par le fait que ce protocole est récent et nous avons trouvé des études récentes sur celui-ci ou des chercheurs ont en évalué les performances en termes de taux de réception des paquets (PRR, *Packet Reception Rate*) dans un environnement fixe.

Dans notre implémentation, nous avons testé le SDN-WISE sous le simulateur COOJA et nous avons évalué les performances de ce protocole en le comparant avec RPL dans un environnement fixe et un autre mobile. Les résultats de nos simulations démontrent que SDN-WISE et mieux que RPL en termes de consommation d'énergie et de taux de réception de paquets dans un environnement mobile.

Au bout de ce travail nous avons approfondi nos connaissances dans le domaine des SDN aussi bien que dans celui des LLNs et nous avons découvert une nouvelle plateforme de simulation qui est COOJA qui nous a permis de faire des dans un environnement fixe et un autre mobile.

Ce travail peut être étendu de plusieurs façons : tester la scalabilité à RPL en prenant en compte : le nombre de nœuds, le nombre de flux, etc ; Comparer SDN-WISE avec LOADng qui est un protocole réactif ; etc.

Nous espérons que ce travail puisse apporter de nouveaux résultats d'évaluation en ce qui concerne le protocole SDN-WISE et le domaine des SDN dans l'IoT.

Annexe 1

I. Environnement de travail

Pour l'implémentation de notre travail, nous avons choisi un environnement de simulation adéquat au réseau sans fils conçu spécialement pour les réseaux de capteurs. Contiki OS 3.0 qui est une distribution linux, accompagné du simulateur COOJA nous a permis de mettre en œuvre notre travail. COOJA est un simulateur de réseau qui permet l'émulation de plates-formes matérielles réelles. C'est l'application de Contiki OS qui se concentre sur le comportement du réseau. Le simulateur est capable de simuler un réseau de capteurs sans fil avec n'importe type de nœud, il permet d'établir les différentes topologies du réseau sans fils et de faire des statistiques sur le comportement des nœuds et du réseau en général comme la consommation d'énergie, la perte des paquets, la visualisation du payload etc. Ce simulateur est aussi équipé d'outils ou plugin, nécessaires pour évaluer les solutions implémentées, comme l'outil de mobilité, le mote duty cycle (Power Tracker), Radio Messages, Mote Output, Simulation Time Line etc.

Notre travail principale était une **Proposition et Implémentation d'un Protocole de routage basé sur les SDNs pour l'internet des objets,** SDN-WISE vu en détail dans le chapitre précédent, est la meilleur solution qu'on a pu implémenter, le lien suivant représente la source qui nous a aidé à implémenter le protocole sur Cooja. [33]

Le tableau 4.1 résume les caractéristiques techniques de SDN-WISE et du protocole RPL:

	SDN-WISE	**RPL**
Année	Avril 2015	Proposé en 2011
Type du réseau	Réseaux LLNs	Réseaux LLN
Couche de transport	UDP (TCP pour le sink et le contrôleur)	UDP
Couche routage	Dijkstra	RPL Routing
Couche internet	IPv6/ 6LoWPAN	6LoWPAN
Couche MAC	CSMA/CA	CSMA/CA
Couche physique	802.15.4	802.15.4

Caractéristique techniques de SDN-WISE et RPL

II. Aperçu sur COOJA:

Cooja (Contiki Os Java Simulation) est un simulateur dédié aux réseaux capteurs, installé sur le système d'exploitation Contiki, Il permet l'émulation avec des plateformes qui existent en réalité comme Skymote et Z1mote. Cooja permet de faire une simulation selon plusieurs paramètres, qui donne des résultats variant ce qui permet d'évaluer le travail.

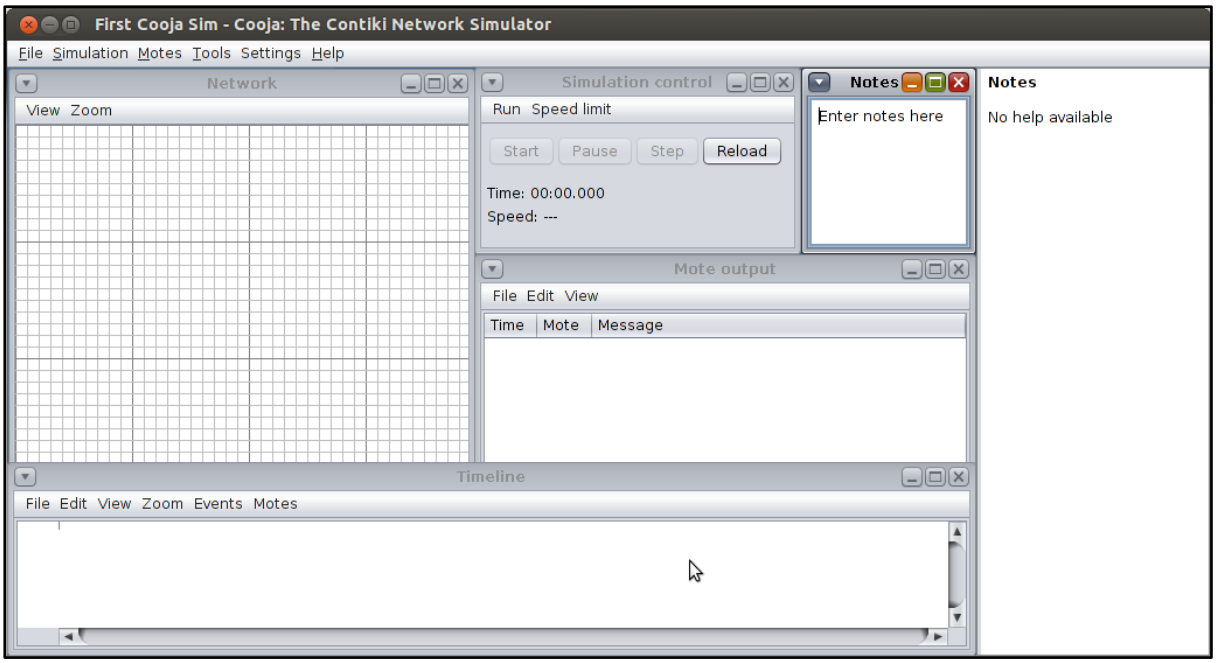

Figure4.1- L'interface de COOJA

La fenêtre Network : elle permet de visualiser la topologie de la simulation ainsi que l'état des nœuds (mobiles ou fixes), la spécification de leur type, leur identification dans le réseau et aussi le champ de couverture de chaque nœud.

La fenêtre Mote Output: c'est l'endroit où les résultats de la simulation sont affichés ainsi que les différents messages échangés entre les nœuds.

La fenêtre Timeline : c'est la fenêtre où on peut voir les transmissions, réception entre les nœuds ainsi que leur état par rapport au temps.

La fenêtre Note : elle permet à l'utilisateur de noter quoi que ce soit sur la simulation.

Annexe 2

I. Installation des outils nécessaires :

D'abord, après avoir téléchargé l'équipement de travail, l'installation du système d'exploitation Contiki 3.0 [34] a été faite sur Virtualbox sous Windows (Logiciel pour créer des machines virtuelles sur ordinateur).

Voici les différentes étapes que nous avons pu suivre afin d'installer l'équipement :

1-Installation de Contiki OS sur Virtual Box

Lien du téléchargement:
https://sourceforge.net/projects/contiki/files/Instant%20Contiki/Instant%20Contiki%203.0/InstantContiki3.0.zip/download.

2- Mise à jour de JAVA à la version 8

SDN-WISE a besoin de java 8 pour s'exécuter, les commandes suivante sont introduite dans le terminal pour cette étape:

```
sudo add-apt-repository ppa:openjdk-r/ppa -y
sudo apt-get update
sudo apt-get install openjdk-8-jdk ant maven2 -y
sudo update-java-alternatives -s java-1.8.0-openjdk-i386
```

3-Mise à jour du path

Commande:

```
sed -i '/export JAVA_HOME=\/usr\/lib\/jvm\/java-7-openjdk-i386/c\export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-i386' ~/.bashrcsource ~/.bashrc
```

4-Installation et compilation de SDN-WISE et COOJA

Commandes:

```
cd ~/contiki/tools
wget https://sdnwiselab.github.io/tools/sdn-wise_java.tar.gz
tar xvf sdn-wise_java.tar.gz
rm sdn-wise_java.tar.gz
cd cooja
git submodule update --init
ant jar_cooja
cd examples/sdn-wise_java
ant compile
```

5-Installation et compilation du contrôleur

Commandes:

```
wget http://sdn-wise.dieei.unict.it/tools/01-GetStarted.tar.gz
tar xvf 01-GetStarted.tar.gz
```

```
rm 01-GetStarted.tar.gz
cd 01-GetStarted
mvn package
java -jar target/01-GetStarted.jar
```

Remarque: au début nous avons eu quelques difficultés lors du téléchargement du contrôleur car le lien du téléchargement a été mis à jour pour accepter que des requêtes https, donc nous avons fait quelques modifications dans le fichier Project Object Model POM.xml.

II. Code source

1. **Contrôleur :** La partie du contrôleur est implémenté dans un fichier nommé Get started, écrit en JAVA le contrôleur déjà expliqué dans les chapitres précédents comporte un fichier **config.ini** dont toutes les configurations requises sont mis en place comme la socket (adresse et numéro de port), le type d'algorithme qui va être utilisé (dans ce cas Dijkstra), le payload max etc.

config.ini ×

```
{
  "Controller": {
    "lower": {
      "PORT": "9991",
      "IP": "localhost",
      "MAX_PAYLOAD": "8192",
      "TYPE": "TCP",
      "IS_SERVER":"TRUE"
    },
    "algorithm": {
      "TYPE": "DIJKSTRA"
    },
    "map": {
      "TIMEOUT": "30",
      "RSSI_RESOLUTION": "20",
      "GRAPH":"CLI"
    }
  }
}
```

- StartController : le rôle principale de cette fonction est d'initialiser le contrôleur en chargeant le fichier config que sont lien est pris en paramètres puis le démarrer.

```
public Controller startController(String configFilePath) {
    InputStream configFileURI = null;
    if (configFilePath == null || configFilePath.isEmpty()) {
        configFileURI = this.getClass().getResourceAsStream("/config.ini");
    } else {
        try {
            configFileURI = new FileInputStream(configFilePath);
        } catch (FileNotFoundException ex) {
            Logger.getLogger(SdnWise.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    Configurator conf = Configurator.load(configFileURI);
    controller = new ControllerFactory().getController(conf.getController());
    new Thread(controller).start();
    return controller;
}
```

- *StartExample:* Cette fonction elle fait référence à StartController dès que le contrôleur est initialisé, le terminale affiche un message "SDN-WISE Controller running….", avant de commencer le traitement le contrôleur attends 60 secondes pour que les nœud dans Cooja seront bien mis en place et que la topologie est faite, après il commence à construire le paquet en initialisant la source qui est évidemment le sink et la destination qui est le nœud concerné, il ajoute un payload au paquet qui représente un message "Hello World" qui sera plus tard acheminer pour arrivé a la destination. Après chaque opération, il attend 2 secondes pour renvoyer un autre paquet à une autre destination.

```
public void startExample() {
    controller = startController("");

    System.out.println("SDN-WISE Controller running....");

    // We wait for the network to start
    try {
        Thread.sleep(60000);

        // Then we query the nodes
        while (true){
            for (int i = 1; i < 12; i++){
                System.out.println("- quering node " + i);
                int netId = 1;
                NodeAddress dst = new NodeAddress(i);
                NodeAddress src = new NodeAddress(1);

                DataPacket p = new DataPacket(netId,src,dst);
                p.setNxhop(src);
                p.setPayload("Hello World!".getBytes(Charset.forName("UTF-8")));
                controller.sendNetworkPacket(p);
                Thread.sleep(2000);
```

- *ManageRoutingRequest:* Pour tous qui est routage le protocole utilise l'algorithme Dijkstra pour déterminer le meilleur chemin. la fonction prend en paramètres les données comme l'adresse S/D et calcule le path et l'envoi au sink qui va acheminer le paquet à son tour.

```
public final void manageRoutingRequest(NetworkPacket data) {

    String destination = data.getNetId() + "." + data.getDst();
    String source = data.getNetId() + "." + data.getSrc();

    if (!source.equals(destination)) {

        Node sourceNode = networkGraph.getNode(source);
        Node destinationNode = networkGraph.getNode(destination);
        LinkedList<NodeAddress> path = null;

        if (sourceNode != null && destinationNode != null) {

            if (!lastSource.equals(source) || lastModification !=
etworkGraph.getLastModification()) {
                results.clear();
                dijkstra.init(networkGraph.getGraph());
                dijkstra.setSource(networkGraph.getNode(source));
                dijkstra.compute();
                lastSource = source;
                lastModification = networkGraph.getLastModification();
            } else {
```

2. Le nœud :

- *initSdnWise*: cette fonction permet de paramétrer chaque nœuds ajoutés à la simulation en initialisant sa valeur qui représente la distance du sink à TTL max +1, le RSSI du sink a 0 et le sémaphore a 0 qui indique que le nœud peut envoyer un message. Toutes ces valeurs sont considérées comme des valeurs par défaut pour chaque nœud ajouté sauf le sink qui prend des valeurs inverses.

```
@Override
public final void initSdnWise() {
    super.initSdnWise();
    commonConstructor();
    setDistanceFromSink(0);
    setRssiSink(255);
    this.setSemaphore(1);
}
```

- *SDN_WISE_Callback:* cette fonction prend en paramètres le paquet reçu du sink, si le paquet est reçu, la première des choses le payload sera affiché dans la fenêtre Mote Output de COOJA grâce au **log**, ensuite une fonction Run Flow Match prend en charge le paquet (détaillé dans le prochain point). Si aucun paquet n'est reçu, rien ne sera affiché.

```
public void SDN_WISE_Callback(DataPacket packet) {
    if (this.functions.get(1) == null) {
        log(new String(packet.getPayload(),Charset.forName("UTF-8")));
        packet.setSrc(addr)
                .setDst(getActualSinkAddress())
                .setTtl((byte) ttl_max);
        runFlowMatch(packet);
    } else {
        this.functions.get(1).function(adcRegister,
                flowTable,
                neighborTable,
                statusRegister,
                acceptedId,
                flowTableQueue,
                txQueue,
                0,
                0,
                0,
                packet);
    }
```

- *Run_Flow_Match:* C'est la fonction qui va traiter le paquet reçu, elle résume tous qu'on a traités dans la partie explicative de la table de flux du chapitre 3. Au début, s'il contient une nouvelle règle cette dernière est ajouté à la table de flux puis le nœud nous informe que la règle a été bien ajouté en affichant : Matched Rule # n° et le contenu de la ligne actuelle qui correspond à cette règle. ensuite si la règle contient une action (Forward to, Turn off radio, Modify etc.), cette dernière serait exécutée. Finalement, le compteur est incrémenté.

```
public final void runFlowMatch(NetworkPacket packet) {
    int j, i, found = 0;
    for (j = 0; j < SDN_WISE_RLS_MAX; j++) {
        i = getActualFlowIndex(j);
        if (matchRule(flowTable.get(i), packet) == 1) {
            log("Matched Rule #" + j + " " + flowTable.get(i).toString());
            found = 1;
            for (AbstractAction a : flowTable.get(i).getActions()) {
                runAction(a, packet);
            }
            flowTable.get(i).getStats()
                    .setCounter(flowTable.get(i).getStats().getCounter() + 1);
            break;
        }
    }
```

3. **Le sink :**

- *initSdnWise :* permet de paramétrer le Sink ajouté à la simulation en initialisant sa valeur qui représente la distance du sink évidemment a 0 car c'est lui même et Le RSSI du sink a 255. Toutes ces valeurs sont considérées comme des valeurs par défaut pour le sink sauf les autres nœuds qui prennent des valeurs inverses.

```
@Override
public final void initSdnWise() {
    super.initSdnWise();
    commonConstructor();
    setDistanceFromSink(0);
    setRssiSink(255);
    this.setSemaphore(1);
}
```

- *ControllerTX*: Grâce à cette fonction le sink peut communiquer avec le contrôleur, cette fonction permet de générer un nouveau paquet dans le cas où la connexion a été établie entre le contrôleur et le sink.

```
@Override
public final void controllerTX(NetworkPacket pck) {
    try {
        if (tcpSocket != null) {
            inviaOBJ = new DataOutputStream(tcpSocket.getOutputStream());
            inviaOBJ.write(pck.toByteArray());
            log("C-TX " + pck);
        }
    } catch (IOException ex) {
        log(ex.getLocalizedMessage());
    }
}

@Override
public void SDN_WISE_Callback(DataPacket packet) {
    controllerTX(packet);
}
```

Run**:** C’est ici que l’objet reçu du contrôleur est traité, dès que le sink reçoit un objet il vérifie si sa longueur est supérieur à 0 c’est à dire il contient des données, le sink affiche que Network paquet a été bien reçu et il l'ajoute à la queue pour pouvoir le transmettre dans le réseau.

```
private class TcpListener implements Runnable {
    @Override
    public void run() {
        try {
            riceviOBJ = new DataInputStream(tcpSocket.getInputStream());
            while (true) {
                int len = riceviOBJ.read();
                if (len > 0) {
                    byte[] packet = new byte[len];
                    packet[0] = (byte) len;
                    riceviOBJ.read(packet, 1, len - 1);
                    NetworkPacket np = new NetworkPacket(packet);
                    log("C-RX " + np);
                    flowTableQueue.put(np);
                }
            }
        } catch (IOException | InterruptedException ex) {
            Logger.getLogger(Sink.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

Bibliographie

[1]. L. Atzori, A. Iera and G. Morabito, "The internet of things: A survey", Comput. Netw., vol. 54, no. 15, pp. 2787-2805, 2010

[2]. H. Schaffers, N. Komninos, M. Pallot, B. Trousse, M. Nilsson and A. Oliveira,"Smart cities and the future internet: Towards cooperation frameworks for open innovation", The Future Internet Lect. Notes Comput. Sci., vol. 6656, pp. 431-446, 2011.

[3]. Shiverware.com. (2019) ”Internet of Things vs Wireless Sensor Networks”. Access date: April 2021, https://shiverware.com/iot/iot-vs-wsn.html.

[4]. JP.Vasseur. The ROLL (*Routing Over Low-Power and Lossy*) terminology document “RFC 7102”. “Terms Used in Routing for Low-Power and Lossy Networks”. January 2014

[5].www.cisco.com "Routing Protocol for LLN (RPL) Configuration Guide, Cisco IOS Release 15M&T. Access date: April 2021.

[6]. Herberg, U., & Clausen, T. (2011). A comparative performance study of the routing protocols LOAD and RPL with bi-directional traffic in low-power and lossy networks (LLN). Proceedings of the 8th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks - PE-WASUN ’11. doi:10.1145/2069063.2069076

[7]. Kharrufa, H., Al-Kashoash, H. A. A., & Kemp, A. H. (2019). RPL-Based Routing Protocols in IoT Applications: A Review. IEEE Sensors Journal, 19(15), 5952–5967. doi:10.1109/jsen.2019.2910881

[8]. Ee, G, K. , Ng, C. K. , Noordin, N. K. , & Ali, B. M.(2010). A Review of 6LoWPAN Routing Protocols. Proceedings of the Asia-Pacific Advanced Network, 30 (0), 71. Doi: 10.7125/apan.30.11

[9]. Kushalnagar, N., Montenegro, G.E., & Schumacher, C.P. (2007). IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. *RFC, 4919*, 1-12.

[10]. Mr.Kameche Abdellah 6LoWPAN Cours Informatiques Embarqués Université Saad Dahleb Blida 2020/2021.

[11]. Khan, M. M., Lodhi, M. A., Rehman, A., Khan, A., & Hussain, F. B. (2016). Sink-to-Sink Coordination Framework Using RPL: Routing Protocol for Low Power and Lossy Networks. Journal of Sensors, 2016, 1–11. doi:10.1155/2016/2635429

[12]. Kharrufa, H, Al-Kashoash, H, Al-Nidawi, Y et al. (2 more authors) (2017) Dynamic RPL for Multi-hop Routing in IoT Applications. In: IEEE Proceedings of WONS 2017. 13th Annual Conference on Wireless On-demand Network Systems and Services (WONS) 2017, 21-24Feb 2017, Jackson Hole, Wyoming, USA. IEEE , pp. 100-103. ISBN 978-3-901882-88-3 https://doi.org/10.1109/WONS.2017.7888753

[13].Jad Nassar, Nicolas Gouvy, Nathalie Mitton. Fonction objectif pour un RPL adapté aux Smart Grids.ALGOTEL 2017 - 19èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications, May 2017, Quiberon, France. pp.4. (hal-01513187)

[14].Patrick Olivier KAMGUEU,"Configuration Dynamique et Routage pour l’Internet des Objets", Pages:16, 21, 24, 25, 26 Doctorat de l’Université de Lorraine (France) et de l’Université de Yaounde 1 (Cameroun),

[15].H. Kharrufa, H. A. A. Al-Kashoash and A. H. Kemp, "RPL-Based Routing Protocols in IoT Applications: A Review," in IEEE Sensors Journal, vol. 19, no. 15, pp. 5952-5967, 1 Aug.1, 2019, doi: 10.1109/JSEN.2019.2910881.
[16].Kamal Benzekki, Abdeslam El Fergougui and Abdelbaki Elbelrhiti Elalaoui"Software-defined networking (SDN): a survey Published online 7 February 2017 in Wiley Online Library"Laboratory of Computer Networks and Systems, Department of Mathematics and Computer Science, Faculty of Sciences, MoulayIsmail University, Meknes, Morocco
[17]. Pliatsios, Dimitrios & Sarigiannidis, Panagiotis & Goudos, Sotirios & Karagiannidis, George. (2018). Realizing 5G vision through Cloud RAN: technologies, challenges, and trends. EURASIP Journal on Wireless Communications and Networking. 2018. 10.1186/s13638-018-1142-1.
[18].Abigail O. Jefia, Segun I. Popoola and Aderemi A. Atayero Department of Electrical and Information Engineering Covenant University, "Software-Defined Networking: Current Trends, Challenges, and Future Directions",Poceedings of the International Conference on Industrial Engineering and Operations Management Washington DC, USA, September 27-29, 2018
[19].I. E. Korbi, M. Ben Brahim, C. Adjih and L. A. Saidane, "Mobility Enhanced RPL for Wireless Sensor Networks," 2012 Third International Conference on The Network of the Future (NOF), 2012, pp. 1-8, doi: 10.1109/NOF.2012.6463993.
[20].Sahrish Khan Tayyaba,Munam Ali Shah,Omair Ahmad Khan,Abdul Wahab Ahmed Department of Computer Science COMSATS Institute of Information Technology, Islamabad, Pakistan.Software Defined Network (SDN) Based Internet of Things (IoT): A Road Ahead July 2017,DOI: 10.1145/3102304.3102319.
[21].Michael Baddeley*, Reza Nejabati*, George Oikonomou*, Mahesh Sooriyabandara†, Dimitra Simeonidou**High Performance Networks Group, University of Bristol, Bristol, United Kingdom.Evolving SDN for Low-Power IoT Networks arXiv:1809.07296v3 [cs.NI] 29 May 2019
[22]. “Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update2015–2020 White Paper,” Cisco.
[23]. SDN-WISE The stateful Software Defined Networking solution for the Internet of Things https://sdnwiselab.github.io/
[24]. Galluccio, L., Milardo, S., Morabito, G., & Palazzo, S. (2015, April). SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks. In 2015 IEEE Conference on Computer Communications (INFOCOM) (pp. 513-521). IEEE
[25].Ndiaye, M., Hancke, G. P., & Abu-Mahfouz, A. M. (2017). Software defined networking for improved wirelesssensor network management: A survey. Sensors, 17(5), 1031.
[26].International Conference on Information Networking –2018January 10th, 2018 -Chiang Mai, Thailand.
[27].Bai, Fan, and Ahmed Helmy. "A survey of mobility models." Wireless Adhoc Networks. University of Southern California, USA 206 (2004): 147.
[28].M. Stehlík, “Comparison of Simulators for Wireless Sensor Networks,” pp. 1–92, 2011.

[29]. [Online]. Available: https://slogix.in/difference-between-skymote-and-z1mote-and-wismote-in-contiki-cooja simulation. [Accessed: 2019].
[30]. BAGULA, B. A. et ERASMUS, Zenville. IoT emulation with Cooja. In: ICTP-IoT workshop. 2015.
[31]. LISCANO, Ramiro, *et al.* Performance Evaluation of SDN-WISE Against RPL-Based Ad-Hoc Wireless Sensor Network Using the Cooja Simulator. In: *International Conference on Wireless Intelligent and Distributed Environment for Communication*. Springer, Cham, 2020. p. 31-39.
[32] Gudipati, A., Perry, D., Li, L. E., & Katti, S. (2013). SoftRAN. Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking - HotSDN '13. doi:10.1145/2491185.2491207
[33] https://sdnwiselab.github.io