



EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité Informatique et Réseaux

présentée et soutenue publiquement par

Remy LEONE

24 décembre 2015

Passerelles  elligentes pour réseaux de capteurs

Directeur de thèse : **Dr. Jean Louis ROUGIER**

Co-encadrement de la thèse : **Dr. Vania CONAN**

Jury

Dr. Jean Louis ROUGIER, Professeur, Télécom ParisTech, France

Dr. Vania CONAN, Thales Communications & Security

Dr. Paolo MEDAGLIANI, Huawei Technologies Co. Ltd.

Dr. Jeremie LEGUAY, Huawei Technologies Co. Ltd.

Dr. Claude CHAUDET, Maître de Conférences, Télécom ParisTech, France

M. HODOR, Hodor

M. Chuck NORRIS, Titre, Unité de recherche, Ecole

Mme Margaret THATCHER, Titre, Unité de recherche, Ecole

Directeur de thèse

Directeur de thèse

Encadrant de thèse

Encadrant de thèse

Encadrant de thèse

Hodor

Last Judge

Examinatrice farouche

TELECOM ParisTech

école de l'Institut Mines-Télécom - membre de ParisTech

46 rue Barrault 75013 Paris - (+33) 1 45 81 77 77 - www.telecom-paristech.fr

A beato torello.

Abstract

Low-Power and Lossy Networks (LLN) are constrained networks typically used in building automation scenarios. To connect natively to the Internet, those networks use a gateway. Because of its key location, at the border of both constrained and conventional networks, it can host advanced features. This thesis offers 3 contributions about those problematics and the research methodologies that come with them.

First we offer an adaptive cache that can adapt lifetime of a resource inside a cache in function of the network state and incoming traffic. Proxy cache are used to speed up request treatment. It can be extended by changing lifetime of a resource inside the cache to regulate the amount of request that the LLN have to handle. The optimal lifetime are solution to multi-objective problems. We propose a method based on genetic algorithm to find a set of solution that belong to the Pareto front of the optimal point of configuration.

Second, we propose an estimator of network traffic and energy consumption based on the observation of the network traffic passing through the gateway. Supervision controls and monitors network state. Usually, supervision is done by sending periodic supervision messages. This approach is costly and might be not always possible. By using the network traffic available at the gateway as a base for a model, it's possible to reduce the amount of active supervision needed to monitor a LLN.

Finally, we propose Makesense, a methodology, coupled to tool ecosystem to create an experiment chain on both simulator and testbed from a unique description. We can combine fast development phase with many iteration then deploying the same code on real nodes to have realistic tests. Once a run is done, the same results analysis toolchain are used. Finally this methodology coupled with continuous integration pave the way to guarantee repeatable experiment.

Résumé

Les Low-Power and Lossy Networks (LLN) sont des réseaux contraints par leurs ressources typiquement utilisés dans des scénarios d'automatisation de bâtiments. Pour se connecter nativement à l'Internet, ces réseaux utilisent une passerelle. Grâce à sa situation à la jointure entre le monde contraint et conventionnel, cette passerelle est à un endroit clé pour accueillir des fonctionnalités réseaux avancées. Cette thèse propose trois contributions autour de ces problématiques de fonctionnalités réseau avancées et des méthodes de recherche qui les accompagnent.

Nous proposons un mécanisme de cache adaptatif permettant d'adapter le temps de vie des ressources dans le cache en fonction du trafic entrant et de l'état des nœuds. Les mécanismes de proxy cache sont utilisés pour accélérer le traitement des requêtes en répondant aux requêtes entrantes à la place des nœuds concernés. Cet aspect peut être étendu en faisant varier les temps de vie des ressources afin de réguler les requêtes touchant le réseau contraint en fonction de ses capacités. Les configurations optimales de temps de vie répondent à des optimisations multi-objectifs. Nous proposons une méthode basée sur des algorithmes génétiques pour trouver le front de Pareto des points optimaux de configuration des temps de vie.

En parallèle, nous proposons un mécanisme d'inférence de supervision du trafic réseau dans le réseau contraint par observation du trafic réseau observé au niveau de la passerelle. Les mécanismes de supervision sont utilisés pour contrôler l'état d'un réseau. L'approche courante consiste à envoyer des requêtes régulièrement afin de connaître l'état d'un nœud. Cette approche est coûteuse à l'échelle de nœuds contraints et doit être limitée. En utilisant le trafic réseau observés à la passerelle comme base d'un modèle, il est possible de réduire le nombre de messages explicites afin de réduire l'impact de la supervision sur le réseau contraint.

Enfin, nous proposons Makesense, une méthodologie couplée à un écosystème d'outils permettant de créer une chaîne d'expériences sur banc de test et simulations à partir d'une description unique. Nous pouvons combiner phase de développement rapide avec des simulations puis déployer le même code sur nœuds réels afin d'avoir des tests réalistes et une analyse de résultats communes à ces deux phases. Enfin, la méthodologie de développement associée permet d'assurer la répétabilité des expériences.

Remerciements

Mettre les classiques Je voudrais remercier mes rapporteurs et mes directeurs de thèses ainsi que Claude Chaudet, Jeremie Leguay et Paolo Medagliani pour leur aide et leur encadrement durant cette thèse.

Mettre ceux qui m'ont aidé Grégoire


Mettre les équipes du LINCIS et de TAI

Je voudrais également remercier chaleureusement Thomas, mon colocataire bien aimé et Marc pour leur soutien à travers les épreuves que ces dernières années m'ont apportées.

Je voudrais remercier aussi Paris Montagne pour ces années à cotoyer le quotidien des chercheurs et la destruction complète de toute autocensure. De même je remercie les communautés liées aux logiciels libres et plus généralement à l'Internet. Ma formation n'aurait pas été la même sans les contributions patientes de tous ses gens qui m'ont formé l'esprit.

Table des matières

1	Introduction	1
1.1	Internet of Things (IoT)	2
1.1.1	Applications des Low-Power and Lossy Networks (LLN)s	3
1.1.2	Changements apportés par l'IoT	5
1.2	Architecture générale	7
1.2.1	Nœuds contraints	7
1.2.2	Passerelles	7
1.2.3	Collecteur - Intégration - Action	8
1.3	Passerelle	8
1.3.1	Fonctionnalités basiques	8
1.3.2	Reverse Proxy & Cache	9
1.3.3	Supervision Réseau	9
1.4	Contributions	9
1.5	Plan du manuscrit	10
2	Environnement technique	11
2.1	Couche basses & réseau - IEEE 802.15.4 & 6LoWPAN	12
2.1.1	Couche physique et liaison - IEEE 802.15.4	12
2.1.2	Couche réseau - 6LoWPAN	13
2.1.3	Architecture 6LoWPAN	13
2.1.4	Couche adaptative	13
2.1.5	Gestion des voisins	14
2.2	Protocole de routage - Routing Protocol Layer (RPL)	15
2.2.1	Fonctionnement du protocole	16
2.2.2	Maintenance du Destination-Oriented DAG (DODAG)	17
2.3	Couche applicative - Constrained Application Protocol (CoAP)	18
2.3.1	Architecture REST & Observations	18
2.3.2	Modèle de message	19
2.3.3	Proxy et mise en cache	19
2.4	Systèmes d'exploitation et simulation - Contiki & Cooja	19
2.4.1	Cycles de veille - ContikiMAC	19
2.4.2	Cooja	20
2.5	Conclusion	20

3	Cache adaptatif	21
3.1	Introduction	22
3.1.1	État de l'art	23
3.1.2	Architecture & Implémentation	23
3.2	Performance d'un cache simple	24
3.2.1	Modèle théorique	25
3.2.2	Simulations & Résultats	26
3.2.3	Validation des performances	27
3.3	Performance d'un cache adaptatif	29
3.3.1	Modélisation de la satisfaction utilisateur	29
3.3.2	Analyse énergétique et modélisation du réseau	30
3.4	Optimisation multi-objectifs	34
3.4.1	Justification de la méthode d'optimisation multi-objectifs	34
3.4.2	Formalisation en problème multi-objectif	34
3.4.3	Résultats expérimentaux	35
3.5	Conclusion	37
4	Supervision active & passive	39
4.1	Introduction	41
4.1.1	Justification	42
4.1.2	État de l'art	42
4.1.3	Contribution	43
4.2	Modélisation	44
4.2.1	Consommation énergétique de transmission et réception	45
4.2.2	Strobbing en CSMA/CA	46
4.3	Supervision passive	47
4.3.1	Supervision passive du trafic réseau	48
4.3.2	Résultats expérimentaux - Topologie en chaine	48
4.3.3	Supervision active & passive	51
4.4	Conclusion 	53
5	Expériences automatisées et reproductibles	55
5.1	Makesense	56
5.1.1	Justifications	56
5.1.2	État de l'art & Écosystèmes	57
5.1.3	Architecture	57
5.2	Construction d'une expérience reproductible	59
5.2.1	Fabrication	59
5.2.2	Déploiement - Execution - Déplacement des traces	59
5.2.3	Mise en forme des résultats intermédiaires	60
5.2.4	Analyse des résultats	61
5.2.5	Présentation des résultats	61
5.2.6	Garanties de reproductibilité par intégration continue	62
5.3	Conclusion & Perspectives	63
6	Conclusion	65

A Codes Sources	67
A.1 Fabrication	67
A.2 Déploiement	69
A.3 Parsing	70
A.4 Analyse	71
A.4.1 Filtrage	71
A.4.2 Fonctions agrégées	71
A.4.3 Traitements en masse	71
A.5 Présentation	72
A.6 Intégration continue (Travis-ci)	73
Bibliographie	77

Table des figures

1.1	Internet des objets (IoT)	2
1.2	Défis techniques rencontrés dans le déploiement d'un réseau de capteurs	6
1.3	Schéma de l'architecture usuelle d'un LLN	7
2.1	Comparatifs des piles de protocoles IETF	12
2.2	Architecture 6LoWPAN	14
2.3	Entêtes 6lowpan	15
2.4	Construction d'un DODAG	16
3.1	Architecture du reverse proxy adaptatif	24
3.2	La topologie radio considérée. $N = 12$ noeud placés sur une grille avec la racine comme noeud central.	27
3.3	(a) Cache Hit ratio (h) comme fonction de la durée de vie de cache c_i . Simulation (traits plein) et théorique (pointillés). (b) Durée de vie du LLN comme fonction de la durée de vie dans le cache c_i . $\lambda_i = 2$ (cercles), $\lambda_i = 1$ (carrés), et $\lambda_i = 0.5$ (triangles).	28
3.4	Mécanisme de crossover	35
3.5	Front de Pareto pour le scénario envisagé.	37
4.1	Supervision et sélection de nœuds	41
4.2	Noeuds impactés par l'acheminement d'une trame depuis le nœud E vers le nœud G.	45
4.3	Nombre moyen de tentatives d'envois en fonction de la taille de la trame.	46
4.4	Topologie réseau	49
4.7	Cout estimé dans les scénarios de chaines	51
4.8	Topologie réseau et radio.	52
4.9	Erreur relative pour la topologie réseau avec une supervision active ($T = 25s$).	52
4.10	Erreur relative pour différents intervalles de supervision active.	53
5.1	Cycles de développement	56
5.2	Évaluation des solutions expérimentales	57
5.3	Schéma général des étapes dans Makesense	58

Chapitre 1

Introduction

Prediction is very difficult, especially if it's about the future.

Niels Bohr

Contents

1.1 IoT	2
1.1.1 Applications des LLNs	3
1.1.2 Changements apportés par l'IoT	5
1.2 Architecture générale	7
1.2.1 Nœuds contraints	7
1.2.2 Passerelles	7
1.2.3 Collecteur - Intégration - Action	8
1.3 Passerelle	8
1.3.1 Fonctionnalités basiques	8
1.3.2 Reverse Proxy & Cache	9
1.3.3 Supervision Réseau	9
1.4 Contributions	9
1.5 Plan du manuscrit	10

Internet s'est développé au cours des dernières décennies, partant d'un petit réseau académique pour devenir ubiquitaire et mondial. Pendant que l'Internet des serveurs, routeurs et téléphones mobiles est devenu plus stable, une autre révolution se préparait : celle de l'IoT. Cette tendance désigne la capacité de systèmes embarqués réduits à se connecter nativement en utilisant les mêmes protocoles réseaux que sur Internet. Touchant de multiples domaines, cette tendance se développe dans des domaines variés et hétéroclites.

Ce chapitre présente une introduction à ce phénomène. L'IoT sera présenté dans la section 1.1. Nous présenterons ensuite l'architecture classique de ces réseaux 1.2 en nous intéressant tout particulièrement à l'étude de la passerelle dans la section 1.3, son intérêt, ses fonctionnalités. Les contributions de cette thèse seront présentées dans la section 1.4 et le plan du manuscrit dans la section 1.5.

1.1 IoT

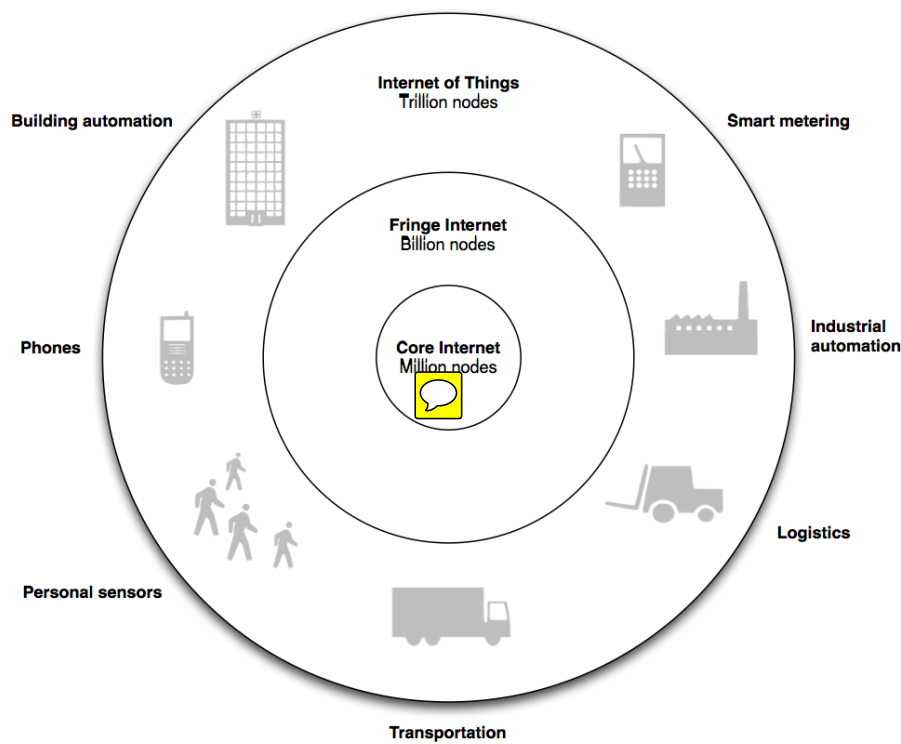


FIGURE 1.1 – Internet des objets (IoT)

Les serveurs et routeurs interconnectés possédant des capacités de traitement importantes composent le cœur de l'Internet [91]. Depuis des décennies, de nombreux autres appareils sont venus s'ajouter à ce cœur originel. La majorité des nœuds composants l'Internet aujourd'hui est situé à la bordure et se compose d'appareils grand public : les téléphones mobiles et les réseaux locaux connectés à l'Internet. La croissance de ces appareils est dépendante du nombre d'utilisateurs et de l'usage que des utilisateurs humains font d'Internet.

L'IoT se construit autour des réseaux pré-existant comme montré sur la figure 1.1 et vient ajouter à des objets des fonctionnalités de connexion utilisant les protocoles de l'Internet. Ces machines sont très diverses comme un capteur, une machine industrielle ou des systèmes de contrôle domotiques. Le nombre de machines et leur versatilité ne feront qu'augmenter [43]. L'essentiel des communications entre ces objets connectés se fera sans l'intervention d'humains. Ils seront intégrés dans des systèmes autonomes. Ce sont des communications désignées sous le terme de Machine to Machine (M2M).

Le M2M désigne les échanges de données entre machines sans l'intervention d'humains. Cette tendance désigne une augmentation continue du trafic réseau entre machines qui s'accélérera à mesure que la complexité du réseau et la variété des appareils augmentera [5]. Les applications de télémétrie (Transmission de données mesurées) sont les plus courantes ainsi que les Supervisory Control and Data Acquisition (SCADA). À l'inverse des solutions de télémétries propriétaire visant des machines et déploiements spécifiques avec des solutions propres et non interopérables, le M2M utilise des protocoles ouverts comme Transport Control Protocol (TCP) et Internet Protocol (IP) [105] pour permettre l'interconnexion entre des machines venant de différents constructeurs.

Les technologies venues du monde IP offrent des standards stables alors que celles vendues par un constructeur offrent moins de garanties d'interopérabilité avec les constructeurs concurrents. Au vu du nombre et de la variété d'appareils ainsi que de leurs constructeurs, l'interopérabilité est un enjeu important. Au lieu de déployer des middlewares complexes, l'utilisation des technologies venues d'Internet permettent d'avoir une intégration entre différents équipements par des protocoles stables.

La réduction des besoins énergétiques [56], de la taille, du prix et du poids des processeurs et des capteurs permet d'ajouter une connectivité sans fil à un grand nombre d'appareils utilisés dans des contextes divers mais avec des ressources énergétiques et de calcul faibles. Ces petites machines peuvent se connecter entre elles pour former des réseaux par exemple dans des bâtiments, des véhicules, et l'environnement [5]. Ces réseaux qui disposent de peu de ressources en calcul et en mémoire. De plus leurs capacités de communications sont limitées. Ainsi on appelle ces réseaux des LLNs.

1.1.1 Applications des LLNs

Les applications des LLNs sont si nombreuses et diverses qu'il est difficile d'en avoir une taxonomie complète [99]. Les cas d'utilisations classiques sont dans les domaines de la surveillance, de la maintenance et de l'optimisation d'installations. Les LLNs sont adaptés aux déploiements à large échelles dans des contextes variés et souvent industriels [43].

1.1.1.1 Applications de grands systèmes

L'objectif des LLNs dans des scénarios de "Smart Metering" consiste à offrir en temps réel des mesures et des relevés sur un grand nombre d'objets avec plus d'aisance de déploiement que ce qu'un SCADA pourrait offrir. Les systèmes classiques fonctionnent le plus souvent par câbles et dans de grands déploiements cette solution peut être difficile. Dans les systèmes où il est nécessaire de surveiller de multiples métriques sur l'acheminement de l'eau du gaz ou de l'électricité, pouvoir avoir des appareils à prix bas, s'installant facilement et pouvant apporter à un centre de commandement des relevés et des mesures en temps réel offre une réelle plus-value. Le terme d'Internet des objets dans ce contexte ne signifie pas forcément rendre accessible sur Internet un barrage hydroélectrique mais de permettre aux machines responsables de surveiller ces équipements de se connecter plus facilement à des réseaux conventionnels.

Mesure intelligente La surveillance des installations situés le long d'une chaîne d'approvisionnement d'énergie est un des cas d'applications les plus communs [37, 45]. L'application consiste à placer des capteurs sur les équipements producteurs et consommateurs d'énergies afin de superviser la production et la distribution de l'énergie. Le but étant de contrôler et surveiller la consommation de ressources telles que l'eau, le gaz ou bien l'électricité. La distribution d'énergie serait alors améliorée car la production suivrait aussi près que possible la demande. Cette surveillance permettrait aussi aux consommateurs de gérer au mieux leur consommation et de la réduire dans certains cas.

Environnements intelligents Ce type de système est mis en place dans les cas de préventions et supervisions de risques environnementaux. Les cas d'utilisations classiques sont la détections des feux de forêts [117], les avalanches et glissements de terrain [101, 2], la détection des risques sismiques et volcaniques [110] ou bien la supervision de la qualité de l'eau et de l'air dans des terrains industriels ou contaminés [54, 79]. La surveillance de l'eau [115] d'une rivière ou nappe phréatique influe sur l'approvisionnement en eau d'une ville. En outre, la surveillance des canalisations afin de détecter les fuites plus rapidement et efficacement est une application intéressante. Le déploiement de ces capteurs permet d'avoir des relevés en temps réel de ces installations. En outre, couplé à des actionneurs, il est possible d'effectuer à distance et automatiquement certaines tâches de maintenance sans solliciter un technicien.

Villes intelligentes Elles sont l'une des applications les plus étudiées des LLN [48, 15]. Bien que le terme n'ait pas une définition canonique, le but de la ville intelligente est de permettre à des systèmes d'être déployés à l'échelle d'une ville afin de faciliter la vie de ses habitants. Les applications concrètes peuvent être multiples. Les systèmes de Smart parking [74] sont un bon exemple de déploiement urbain permettant d'avoir en temps réel la disponibilité des places de parking dans le but de réduire le temps et le carburant utilisé pour trouver une place. Des services de supervision du trafic automobile et des piétons peuvent également adapter la synchronisation des feux de circulations afin d'améliorer la fluidité du trafic [27].

Sécurité et Urgences Des systèmes utilisant des LLNs sont mis en œuvre dans des cas de protection et surveillance de zone. Il peut s'agir de détection une présence de liquide dans des installations électriques à haut risques. De détecter la présence ou les fuites de gaz dangereux dans des usines chimiques ou bien de détecter les risques dus aux radiations dans des centrales. Les systèmes de détection d'intrus autour de zone sécurisés [14] sont également présents. Dans ce genre de situation, les systèmes doivent résister à de nombreuses attaques ou altérations de leur environnement tout en étant capable de répondre en temps réel en cas d'attaque. Ces approches peuvent être couplées avec celle des villes intelligentes dans le cas de surveillances des ponts, des routes (nid de poules) ou de grandes structures par leurs vibrations [55].

Ces déploiements sont sous des contraintes énergétiques strictes et doivent rapporter en temps réel les événements tout en s'acclimatant à des conditions rudes de déploiements. En outre, l'interopérabilité de ces systèmes complexes et très hétérogènes les uns avec l'autre est une nécessité afin d'offrir une valeur ajoutée par rapport aux systèmes dits "verticaux" où un seul constructeur offre une solution complète [98].

1.1.1.2 Applications industrielles

Dans des scénarios industriels, il est courant de trouver des SCADA permettant d'avoir une vue complète de l'état d'une machine ou d'un processus industriel. Cependant là où les interconnexions entre ces SCADA se faisait par câble ou bien par des middlewares propriétaires complexes et limités, on assiste désormais à des communications M2M reposant sur des standards réseaux compatibles avec ceux utilisés sur Internet [98].

Les applications industrielles des LLNs sont nombreuses. Même si l'informatique industrielle s'est grandement développée, le fait de pouvoir communiquer en temps réel avec une granularité très fine sur l'ensemble des machines en utilisant les mêmes protocoles interopérables que sur Internet est réellement innovante. Ici les LLN utilisent des protocoles interopérables et des standards entre tous les équipements permettant d'avoir toutes les machines sur le même plan que les équipements connectés au Local Area Network (LAN) usuel.

Agriculture et élevage intelligent Les exploitations agricoles [108, 95] peuvent utiliser les LLNs surveiller l'état des plantations en terme d'irrigation, la présence de pesticide ou produits chimiques, l'acidité des sols et les conditions de météorologiques. D'autres systèmes peuvent être utilisés pour surveiller l'état de santé d'un animal, s'assurer que son état est conforme avec les normes en vigueur et s'assurer de sa traçabilité [106] ainsi que sa localisation.

Gestion de bâtiment - Domotique La surveillance d'un bâtiment [72, 35] est un cas d'utilisation courant pour les LLNs. Le but est d'obtenir au sein d'un bâtiment un système pouvant superviser l'état de l'immeuble sur différents critères comme le chauffage, l'air conditionné, la ventilation et l'alumage des pièces, la fermeture des portes ou la détection de cambriolage[71]. Une fois ce contrôle disponible, il est possible de contrôler le chauffage de manière beaucoup plus automatisée sans l'intervention d'un technicien.

L'intégration de ces capteurs très hétérogènes est un défi de même que le contrôle de logique des actionneurs qui en découle. Les connexions sont souvent intermittentes et sont intégrés avec de

grands systèmes industriels. Les communications M2M et l'interopérabilité de ces systèmes issus de différents constructeurs est essentiel. Ces systèmes de surveillance sont de plus déployés dans des environnements difficiles où les contraintes de sécurité sont importantes ainsi que des besoins de temps réel.

1.1.2 Changements apportés par l'IoT

La réussite de l'IoT dépend de nombreux éléments parmi lesquels se trouve l'intégration de ces données nombreuses et variées dans des modèles de services cohérents et proposant une réelle valeur ajoutée. Les innovations technologiques les plus cruciales sont axées sur la réduction de la consommation énergétique, l'accord de constructeurs autour de protocoles interopérables.

1.1.2.1 Trafic

Profils de trafic sporadiques Dans les réseaux conventionnels, les applications de messageries, la consultation de page web et les services de vidéos à la demande constituent une large part du trafic. Ce type de trafic est asymétrique (les clients consultent plus qu'ils ne publient), volatile et un volume important d'information est échangé lors de chaque session. Le trafic typique d'un LLN sera diamétralement opposé au précédent. Les informations échangées sont modestes, beaucoup plus régulières et les fréquences d'envoi seront également faibles. Les messages sont collectés et interprétés ailleurs dans le réseau et les informations manquantes seront ignorées ou bien extrapolées.

Contraintes de connexions bruitées Dans les réseaux conventionnels, les médiums de communications ont peu de pertes [91] qu'il s'agisse de fibre optiques ou bien cuivrés. À la bordure des réseaux, la vaste majorité des appareils qui formeront les LLNs seront connectés de manière intermittente et inconsistante. Ils sont en sommeil afin d'économiser leurs batteries et leur connexion sans-fil aura une faible bande-passante. Les protocoles de transport traditionnels comme TCP gèrent les pertes de paquets en envoyant de nouveau des paquets. Même si peu de données transitent, l'inefficacité du renvoi d'informations ainsi que la gestion des différentes connexions générerait une redondance non nécessaire. La perte d'une information dans le cas de LLN est souvent acceptable ainsi des gestions plus permissives des pertes sont à recommander pour ces usages.

1.1.2.2 Variétés des usages

Standards variés Les constructeurs d'équipements qui formeront des LLN sont nombreux et le champ d'action du M2M est suffisamment grand pour avoir plusieurs visions concurrentes en activité. La croissance de l'IoT est attendue en utilisant une suite d'opportunités diverses qui en fonctionnant avec des systèmes interopérables peuvent créer une valeur ajoutée [5]. L'IoT nécessite coopération, partage de données et des standards interopérables. Des organismes nationaux et internationaux se penchent sur les questions de standard d'interopérabilité : Allseen Alliance, Industrial Internet Consortium, Open Interconnect, Thread, IPSO Alliance, IEEE. Cependant la tâche est complexe. Différents segments (grand public, industriel), avec différentes chaînes de valeur, différents délais de standardisation (time to market), taille du segment et l'influence régionale font que de nombreux standards coexistent. Chacun de ses organismes essayant d'obtenir autant d'influence que possible.

Variétés des appareils et des contraintes L'un des aspects mis en avant des LLNs seront leurs étendue et leur diversité qui ne fera que croître. Cette croissance appelle de nombreux défis en termes de compatibilité et de maintenance des systèmes. [92]. Cette fertilité pose les problèmes d'interopérabilité et standardisation aux différents acteurs [5, 8].

Cycle de vie Le déploiement n'est qu'une étape de la vie d'un appareil contraint. Les fonctions d'un nœud contraint peuvent changer et nécessiter des mises à jour logicielles. En raison de déploiements physiques il peut être difficile de mettre à jour et des solutions de migrations, maintenance

et déploiement d'applications sont requises [12, 100]. Ainsi développer des solutions assurant une rétrocompatibilité est important.

Indiquer qu'il faut gérer les problématiques de mises à jour de sécurité et de Over The Air (OTA) afin d'avoir une culture dans lequel il est facile de corriger des problèmes.

Là encore, la passerelle a un rôle essentiel de traducteur entre différentes technologies et standards [119]. Les protocoles orientés vers nœuds contraints et utilisant différentes méthodes d'accès physiques devront être convertis vers des paquets interopérables avec les systèmes existants.

1.1.2.3 Défis techniques

Comme nous l'avons vu dans la section 1.1.2, il existe des défis de fond dans la mise en place de ces LLNs. Les défis techniques des LLNs sont aussi nombreux et variés. La figure 1.2 montre les principaux problèmes techniques dans les déploiements classiques de LLNs nous en détaillerons quelques uns.

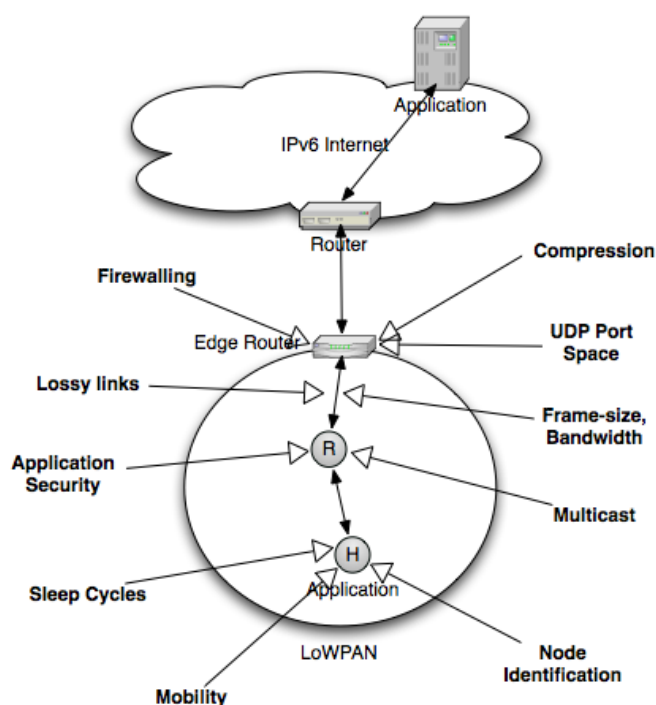


FIGURE 1.2 – Défis techniques rencontrés dans le déploiement d'un réseau de capteurs

Cycles de sommeil Lors de leur période d'inactivité et afin d'économiser autant d'énergie que possible, les nœuds se mettent en sommeil. Durant cette période leur consommation énergétique est inférieure de plusieurs ordres de grandeur à leur consommation moyenne [53]. Cependant lors de ces phases, le nœud est difficilement joignable. En outre, dans le cas de communications asynchrones, il peut être nécessaire de transmettre plusieurs fois un même paquet afin qu'il soit reçu [32]. Dans les cas où les cycles d'endormissement sont gérés par une autorité centrale, il est nécessaire d'introduire de la signalisation supplémentaire afin de gérer les cycles de sommeil [1]. Les attaques au déni de sommeil empêchent les nœuds de dormir et use leur batteries ou leurs ressources déjà limitées pour rendre à terme le réseau indisponible [93].

Liens bruités Les liens étant bruités et instables, les pertes de paquets sont nombreuses. De plus, des délais importants sont courants et dus à une force de signal faible, des collisions, des canaux

sur- utilisés ou plus généralement la congestion d'un nœud [7]. En cas de trop fortes pertes et des conditions réseaux trop instables des nœuds peuvent être amenés à modifier les parents qu'ils ont et les routes utilisées [23].

Écoute passive Dans le cas de communications asynchrones, lorsqu'un nœud reçoit un signal, il doit le décoder et l'analyser afin de décider si cette transmission lui est destinée et si une tâche doit être accomplie. Ce décodage systématique a un coût élevé dans des environnements denses [63] où la couche MAC n'est pas adaptée.

Mobilité La mobilité des nœuds peut entraîner des pertes de connectivité, des reconfigurations de topologie réseau et des congestions [64]. Dans le cas des Body-Area Network (BAN) il peut s'agir de réseaux entiers qui sont mobiles [20]. Des mécanismes de découvertes de voisins et de topologies peuvent permettre de gérer les cas de mobilité au sein d'un même Low-Power Wireless Personal Area Network (LoWPAN) [80].

Sécurité Les problèmes de sécurité sont présents [84]. Que ça soit au niveau du pare-feu (contrôlant les flux entrants et sortants du réseau), du chiffrement de bout en bout des messages ou bien l'identification et l'authentification des nœuds [107]. Mettre les nœuds contraints sur des réseaux IP les expose au même type de risques.

1.2 Architecture générale

Les applications très diverses des LLN. On peut distinguer plusieurs classes de nœuds dans cette architecture [4, 5].

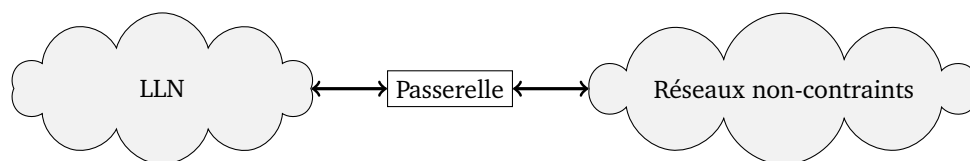


FIGURE 1.3 – Schéma de l'architecture usuelle d'un LLN

Les LLN seront des “stubby networks”. Les données sortent et rentrent du réseau par un seul chemin logique et ne font pas transiter des paquets pour d'autres réseaux.

Nous n'étudierons que les approches “IP”, d'autres approches de type cellulaires sont étudiées (LoRaWAN) et ne seront pas couvertes par ce manuscrit.

1.2.1 Nœuds contraints

A la bordure du réseau, on trouve des appareils simples et souvent contraints. Les améliorations technologiques sont utilisées pour baisser les coûts de production plutôt que pour améliorer les performances [78], ainsi leurs performances resteront modestes. Ils transmettent et reçoivent des trafics de données faibles comme un relevé de capteur. Les plateformes matérielles sont modestes (processeur, mémoire) et la consommation énergétique doit être faible afin qu'ils puissent avoir une grande durée de vie. Pour ce faire, les nœuds se mettent en veille autant que possible. Ces appareils communiquent avec le reste d'un réseau en passant par des passerelles.

1.2.2 Passerelles

Ces passerelles (ou routeur de bordure) ont pour charge de se connecter aux capteurs et autres appareils contraints et de transmettre les informations obtenues à des réseaux tiers. Ces nœuds sont

généralement moins contraints et peuvent disposer de plusieurs moyens de communications ou protocoles (par exemple : CPL, Bluetooth, ZigBee, IEEE 802.15.4 ou Wi-Fi). Afin de gérer de multiples cas de figures, il est important que ces passerelles puissent s'adapter à de multiples protocoles des nœuds contraints et puissent les traduire vers les protocoles classiques. En plus de cette mission de connectivité, elles auront également pour charge d'établir des tables de routage [114], de faire de la découverte de service [22]. Ces fonctionnalités peuvent s'ajouter à des fonctions de contrôle de fréquence de transmission de données ou bien de contrôle de la topologie réseau. Enfin des fonctionnalités réseaux comme un pare-feu, du contrôle d'accès ou de la supervision peuvent également se produire ici.


1.2.3 Collecteur - Intégration - Action

La mise en forme et l'agrégation des informations récoltées se place sur un serveur distant [4, 5]. Bien que les passerelles soient en charge d'organiser les connexions vers les nœuds, l'intégration de ces données vers des services cohérents et à valeur ajoutée se feront au niveau de serveurs distants. Ces services d'intégration de données gèrent des flux de données en provenance de sources multiples. Ces fonctions d'intégrations s'intégreront à d'autres sources de données d'origine diverses telles que des bulletins météo ou des prévisions de trafic routiers [46, 43]. C'est ici qu'on trouve la visualisation des données, la détection d'anomalies et le tri des données. Dans cette architecture, ces fonctions d'intégration seront au plus proche des utilisateurs finaux afin que seuls les alarmes, exceptions ou notifications pertinentes soit envoyées.

1.3 Passerelle

La passerelle offre un point unique qui est suffisamment proche du réseau de capteurs pour avoir une vue informée tout en étant elle-même non contrainte. Elle est nécessaire et incontournable car les nœuds n'ont pas les moyens de communications nécessaires pour se connecter à des réseaux conventionnels en raison de leurs contraintes physiques. Ainsi le rôle de la passerelle est prépondérant pour effectuer une interconnexion efficace.

1.3.1 Fonctionnalités asiques

Rétrocompatibilité  avec l'existant Les nœuds auront des piles protocolaires variées. Afin de se connecter à l'existant, ces passerelles devront communiquer sur les technologies usuelles comme 4G, Wifi et Ethernet. Cependant elles devront aussi pouvoir parler des protocoles adaptés aux nœuds contraints comme le Bluetooth low energy, IEEE 802.15.4. Ainsi il est pertinent de mettre au niveau de la passerelle des fonctionnalités de traductions transparentes de protocoles. Il peut s'agir de protocole réseau comme IPv4/IPv6 qui ont des modes de fonctionnement différents ou bien des fonctionnalités de traductions de protocoles applicatifs tel que HTTP vers des protocoles spécifiques.

Routeur de bordure Dans le cas où les nœuds sont connectés à un autre réseau, la passerelle fait office de routeur de bordure et de pare-feu. Elle échange des routes vers les nœuds de son réseau avec l'extérieur et contrôle les accès aux ressources. Lors du déploiement d'un réseau classique en IPv4 on a une passerelle qui a pour charge d'offrir une connexion à l'Internet. Cependant cette connexion dans les cas usuels de IPv4 repose sur un Network Address Translation (NAT). Or dans le cas où on doit gérer un très grand nombre d'adresses comme c'est prévu dans l'IoT cette approche n'est pas envisageable. Les NAT ralentissent les traitements et ne sont pas pertinents dans le cas où l'on a une technologie comme IPv6 qui permet un très grand nombre d'adresses. Par conséquent les technologies classiques d'interconnexions entre réseaux doivent être mis à jour pour tenir compte des spécificités

de l'IoT. Nous verrons dans la section 2.1.2 les adaptations spécifiques faite à IPv6 pour s'adapter aux scénarios contraints.

1.3.2 Reverse Proxy & Cache

Fonctionnalités de cache Un cache permet d'accélérer l'obtention d'une ressource précédemment demandée. Nous nous intéressons au proxy cache utilisés pour du trafic applicatif comme sur le web [111]. Une ressource est disponible dans le cache pour être utilisée en lieu et place de la ressource réelle pendant un certain temps de vie. Le temps de vie de cache est déclaré usuellement par la ressource même. Cependant ce temps de vie peut être manipulé par le reverse proxy qui peut prendre la décision selon le contexte d'utiliser une ressource pour un temps différent.

Qu'est ce qu'un cache qui est spécifique a l'IoT Les avantages d'un cache sont aussi importante dans le contexte de l'IoT. Un cache peut être utile afin d'éviter qu'un nœud déjà contraint en ressources ne soient obligés de répondre plusieurs fois à une même requête. L'utilisation d'un cache placé stratégiquement permet en outre d'éviter d'aller solliciter un réseau contraint peu rapide et permet d'avoir une latence améliorée. Dans le cas où un nœud sera accessible directement par des utilisateurs non identifiés, le reverse proxy sera une nécessité pour contrôler les requêtes entrantes. Cette utilisation est déjà celle qui en vigueur le web [111].

Les caches permettent aussi de gérer dynamiquement les temps de vie d'une ressource en fonction des conditions d'utilisation du LLN. Ainsi le reverse proxy peut réguler le nombre de requêtes arrivant dans le LLN en modifiant les temps de vie des ressources interrogées.

1.3.3 Supervision Réseau

Présentation La supervision désigne la surveillance continue d'un équipement réseau afin de s'assurer que son fonctionnement et ses performances sont celles attendues [69]. Le cas de supervision le plus courant est une supervision active. Elle utilise des messages explicites pour demander l'état d'un équipement. Typiquement il s'agit d'un nœud qui va périodiquement envoyer une requête pour connaître l'état d'un équipement. Une autre approche de la supervision est dite passive quand elle n'utilise que les informations qui sont déjà disponibles sans aucune autre intervention explicite. Typiquement utilisée dans la supervision réseau, il peut s'agir de remontée de traces de trafic réseau qui sont passées par un routeur de bordure ou bien de sondes avancées. Ces sondes peuvent être mises sur la passerelle afin d'avoir une bonne vue sur les flux réseaux entrants et sortants.

Problématique de supervision dans les LLNs Les techniques de supervisions sont aujourd'hui connues et bien étudiées, cependant elles sont basées sur un jeu d'hypothèses assez fortes sur les nœuds. Dans le cas idéal, ils sont toujours actifs et peuvent répondre aux requêtes de l'administrateur. Cependant dans le cas d'un LLN, les nœuds sont souvent endormis et ne peuvent répondre aussi régulièrement aux requêtes. Par conséquent les techniques usuelles de supervisions sont coûteuses énergétiquement et peu applicable sur de larges LLNs.

1.4 Contributions

Le but de cette thèse est d'exposer comment les fonctionnalités de cache et de supervision peuvent être adaptées aux LLNs. Toutes les techniques et solutions que nous proposons ont pour but d'améliorer les performances et la sûreté d'un réseau de capteur.

Nous proposons tout d'abord un mécanisme de cache adaptatif Nous verrons comment le routeur de bordure peut être utilisé pour héberger des services de mises en cache où le temps de vie d'une information est adapté en fonction de la durée de vie estimée du réseau et du trafic qui y arrive. Une

approche utilisant des algorithmes génétiques sera proposées afin d'avoir un ensemble de solution optimale variée.

Nous proposons ensuite un mécanisme de supervision passif et actif Nous verrons comment à partir de l'observation des paquets passant par le routeur de bordure, il est possible d'inférer une matrice de trafic pour l'ensemble du réseau en utilisant la topologie afin de déduire l'énergie consommée. Nous compléterons cette analyse avec des messages de supervision actifs afin de compléter optionnellement nos prédictions.

Nous proposons ensuite une méthodologie d'expérience reproductible Nous verrons enfin comment organiser une expérience à la fois en simulation et sur nœuds réels afin d'obtenir un mécanisme d'expérience reproductible.

1.5 Plan du manuscrit

Les contributions de cette thèse exposent différentes fonctionnalités et améliorations que la passerelle peut offrir afin d'améliorer les performances et la fiabilité de ces réseaux.

Dans le chapitre 2 nous exposerons les principaux protocoles et systèmes utilisés pendant le reste du manuscrit.

Le chapitre 3 exposera un service de cache adaptatif. Il permet d'adapter la durée de vie d'une information dans le cache afin que ce dernier puisse répondre à la place du nœud à l'origine de cette information. La méthode de résolution et les principaux résultats seront détaillés.

Dans le chapitre 4 nous verrons un service de supervision adapté aux LLNs et permettant d'envoyer en quantité moins importantes des informations de supervision.

Puis nous montrerons dans le chapitre 5 une méthodologie de recherche reproductible appliquée au réseau de capteurs.

Le chapitre 6 apportera une conclusion et ouvrira sur des prolongements possibles de cette thèse

Chapitre 2

Environnement technique



Contents

2.1	Couche basses & réseau - IEEE 802.15.4 & 6LoWPAN	12
2.1.1	Couche physique et liaison - IEEE 802.15.4	12
2.1.2	Couche réseau - 6LoWPAN	13
2.1.3	Architecture 6LoWPAN	13
2.1.4	Couche adaptative	13
2.1.5	Gestion des voisins	14
2.2	Protocole de routage - Routing Protocol Layer (RPL)	15
2.2.1	Fonctionnement du protocole	16
2.2.2	Maintenance du DODAG	17
2.3	Couche applicative - Constrained Application Protocol (CoAP)	18
2.3.1	Architecture REST & Observations	18
2.3.2	Modèle de message	19
2.3.3	Proxy et mise en cache	19
2.4	Systèmes d'exploitation et simulation - Contiki & Cooja	19
2.4.1	Cycles de veille - ContikiMAC	19
2.4.2	Cooja	20
2.5	Conclusion	20

Comme nous l'avons vu dans le chapitre 1, les réseaux de capteurs ont des applications diverses et s'installent sur des plateformes matérielles très variées. La loi de Koomey [56] stipule que la quantité d'opérations processeur pour une quantité d'énergie fixée double tous les 1.57 ans. Cette tendance illustre qu'il est possible d'avoir des processeurs dans des scénarios avec des contraintes élevées.

Dans les cas où aucune contrainte spécifique n'a lieu sur les capteurs, il peut être envisageable d'utiliser une pile protocolaire classique. Cependant dans le cas où les nœuds sont contraints, il est nécessaire d'avoir des protocoles spécifiques et adaptés. L'IETF et l'IEEE contribuent à la construction d'une pile protocolaire adaptés aux réseaux de capteurs. D'autres protocoles de type cellulaire sont également disponibles. Qu'ils soient ouverts comme LoRa ou bien propriétaire comme Sigfox [17], ils sont utilisés pour des trafics faibles et réguliers. Nous avons concentré notre étude sur la pile de protocoles IETF recommandée représentée sur la figure 2.1, car ces protocoles disposent

d'études académiques profondes et des implémentations sont directement disponibles. Il est à noter que d'autres piles protocolaires propriétaires et concurrentes existent et qu'une des missions de la passerelle consiste à offrir des interfaces et des traductions interopérables entre toutes ces piles. Nous étudierons dans cette partie quelques uns de ces protocoles en nous concentrant sur ceux utilisés au cours des chapitres suivants.

La section 2.1 décrira la couche physique et liaison de notre pile. La section 2.1.2 décrira la couche réseau et sera complétée par la section 2.2 qui décrira le protocole de routage que nous avons utilisé. La section 2.3 décrira quant à elle la couche applicative finale. Nous terminerons ce tour d'horizon de notre environnement technique en présentant les fonctionnalités de Contiki et de son environnement de simulation COOJA dans la section 2.4 puis la section 2.5 conclura ce chapitre.

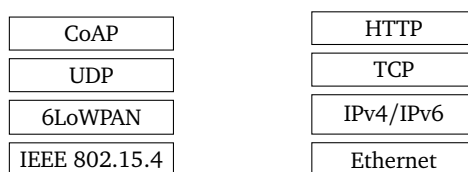


FIGURE 2.1 – Comparatifs des piles de protocoles IETF

2.1 Couche basses & réseau - IEEE 802.15.4 & 6LoWPAN

Dans cette section nous étudierons les protocoles utilisés pour permettre à un LLN de construire une topologie multi-sauts. Nous ne présenterons ici que les points essentiels de ces couches que nous utiliserons dans les travaux futurs.

2.1.1 Couche physique et liaison - IEEE 802.15.4

La couche IEEE 802.15.4 est à la fois une couche physique et liaison dans le modèle OSI [9]. Elle est conçue pour être utilisée dans des appareils aux ressources limitées pour leur permettre d'avoir une connexion sans fil nécessitant peu de complexité et de ressources. Ce protocole appartient à la famille des protocoles Low Rate Wireless Personal Area Network (LRWPAN). Il permet typiquement des portées de communications de l'ordre de 10 à 50 mètres, permet d'avoir un débit de 250 kbits/s. Ce protocole est adapté pour les capteurs passant beaucoup de temps en veille. D'autres protocoles tel que le Bluetooth Low- Energy peuvent être utilisés même si l'interopérabilité de IEEE 802.15.4 avec BLE est toujours une question ouverte [90].

Types de nœuds IEEE 802.15.4 distingue plusieurs types de nœuds, il y a les Full Function Device (FFD) qui peuvent faire transiter des paquets les uns pour les autres et les Reduced Function Device (RFD) qui ont des ressources plus modestes et ne peuvent qu'envoyer des paquets qu'à un autre FFD. Ce mode de fonctionnement est adapté dans le cas de capteurs hétérogènes comme c'est le cas dans notre étude.

Couche MAC La couche Media access control (MAC) peut être utilisée de deux façons : En mode avec balises dans lequel une approche Time Division Multiple Access (TDMA) est possible afin de garantir des délais de transmissions [1]. Ou bien en mode sans balises dans lequel des mécanismes Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) sont utilisés. Dans cette thèse, nous utiliserons la version sans balise qui utilise un mécanisme de type CSMA/CA afin de gérer les collisions éventuelles.

Protocoles compatibles IEEE 802.15.4 est compatible avec de nombreux autres protocoles réseaux industriels. Parmi les plus courants nous pouvons citer Zigbee [3], Wireless HART [65] et 6LoWPAN que nous utiliserons dans nos travaux.

2.1.2 Couche réseau - 6LoWPAN

Le rôle de la couche réseau est de rendre interopérable des liens hétérogènes. A la bordure du réseau de capteurs, se trouvera un routeur qui aura plusieurs connexions avec d'autres réseaux par exemple WiFi, Ethernet ou encore en LTE. Puisque les adresses IPv4 s'épuisent et que les LLNs comportent un grand nombre de nœuds, IPv6 pourrait permettre l'adressage de chaque appareil. Utiliser IPv6 sur les capteurs simplifierait à la fois la connexion des capteurs à l'Internet mais aussi le processus de développement. Cependant, IPv6 n'est pas adapté tel quel à IEEE 802.15.4. De plus, le support des entêtes IPv6 sur du IEEE 802.15.4 laisserait peu de place pour le contenu applicatif. Les bandes passantes faibles, les ressources limitées en énergie et la taille maximale de 127 octets alors que la Maximum transmission unit (MTU) de IPv6 est de 1280 octets sont les contraintes les plus fortes de IEEE 802.15.4. Une couche d'adaptation est requise.

L'Internet Engineering Task Force (IETF) a créé le groupe de travail 6LoWPAN pour définir comment faire fonctionner IPv6 par dessus IEEE 802.15.4. Le groupe de travail avait pour tâche de définir un mécanisme de découverte de voisins adaptés aux LLNs, décrire des mécanismes de compressions d'entêtes et de définir des approches et un cadre pour les protocoles de routages ultérieurs. Le groupe de travail a décrit les hypothèses, problématiques et buts du 6LoWPAN [60] puis a proposé la couche d'adaptation entre IPv6 et IEEE 802.15.4 proposant un format de trame pour la transmission des paquets, une méthode pour définir les adresses en lien local et des configurations d'adresses automatiques, la compression des entêtes et processus de transfert de trames dans les réseaux maillés [76].

Nous avons voulu par le choix de 6LoWPAN privilégier les technologies et les paradigmes provenant de IP. 6LoWPAN [98] est un sujet vaste, nous ne présenterons ici que les aspects relatifs à la gestion de la topologie et à l'adaptation à la passerelle.

2.1.3 Architecture 6LoWPAN

Les LLNs sont présentés comme étant des réseaux "stubby", ils ne font pas transiter du trafic en provenance d'autres réseaux. La Figure 2.2 présente l'architecture mise en avant par 6LoWPAN. Il existe essentiellement trois types de nœuds. Les nœuds (ou 6LNs représentés par un H sur la figure 2.2) dans un LoWPAN utilisent leur capteurs, envoient et reçoivent des messages mais ne transfèrent pas des messages pour le compte d'autres nœuds. Les routeurs (ou 6LRs représentés par un R sur la figure 2.2) sont des nœuds intermédiaires qui font suivre des messages entre leur origine et leur destination. Enfin, les routeurs de bordures (ou 6LBRs) sont la connexion entre des nœuds dans un réseau LoWPAN et d'autres réseaux, par exemple l'Internet. Typiquement, les nœuds et routeurs sont contraints en énergie et en ressource de calcul alors que le routeur de bordure ne l'est pas.

Les 6LoWPAN Border Router (6LBR) sont responsables de la dissémination des préfixes IPv6 et de la compression des entêtes à travers le LoWPAN. Ils maintiennent également un cache de toutes les adresses IPv6 et des identifiants EUI-64 afin de pouvoir effectuer des Duplicate Address Detection (DAD).

Le routeur doit faire office de firewall. on a plus de NAT. Essayer de préciser ce point.

2.1.4 Couche adaptative

IPv6 spécifie que la MTU entre deux hôtes doit être de 1280 octets, or IEEE 802.15.4 ne le permet pas car les trames n'ont que 127 octets. Ainsi IPv6 tel quel ne peut fonctionner. En outre, la couche

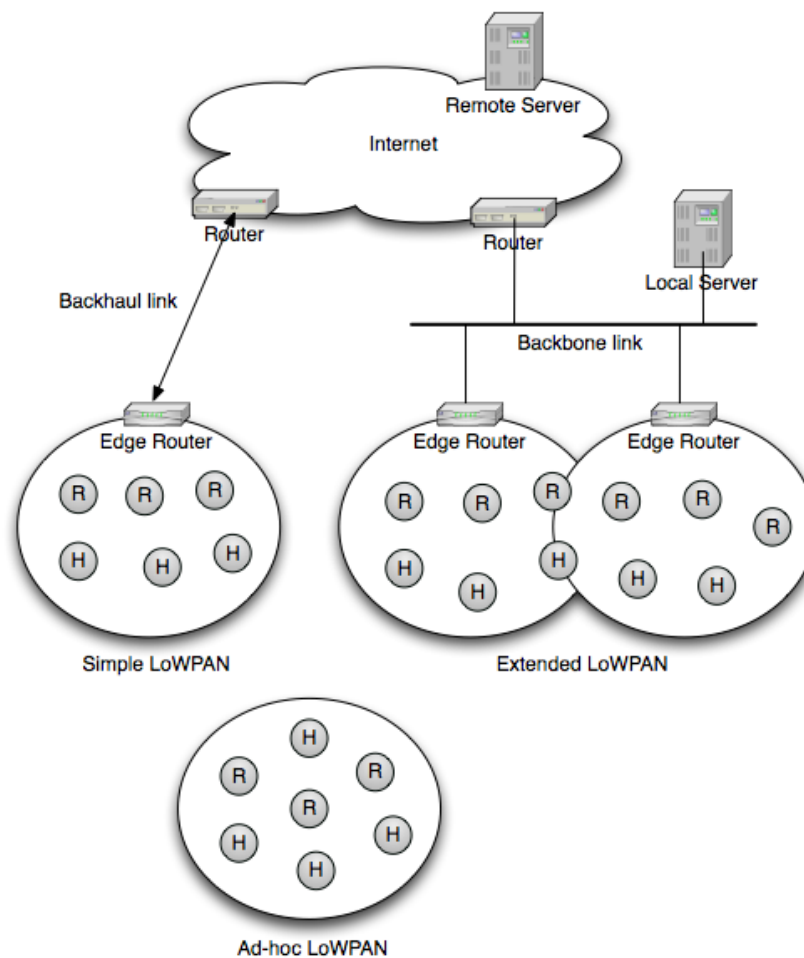


FIGURE 2.2 – Architecture 6LoWPAN

physique prends 25 octets et laisse 102 pour les couches supérieures (elle n'en laisse que 81 dans le cas où des options de sécurité sont actives). Si on laisse IPv6 tel quel, 40 octets sont pris n'en laissant que 41 pour les couches supérieures. UDP qui est un protocole de transport classique en prends 4 à 8 octets ne laissant que 33 octets pour les couches applicatives soit une efficacité modeste de 26 %.

6LoWPAN empile les entête comme IPv6. 4 entêtes sont définies : L'entête d'expédition, l'entête de compression IPv6, l'entête de fragmentation et l'entête de réseau maillé (par défaut, seul les deux premiers sont utilisés). Au début de chaque entête se trouve un identifiant qui indique son format. 6LoWPAN effectue une compression en retirant les prefix IPv6 qui est le même au sein du réseau. Les Interface ID (IID), les adresses de source et de destination sont également compressés.

2.1.5 Gestion des voisins

Dans les réseaux IPv6 classiques, les nœuds et les routeurs se découvrent en utilisant Neighbor Discovery Protocol (NDP). Ce protocole permet d'apprendre les prefixes et les paramètres de configuration liés à la configuration des adresses ; de trouver les routeurs dans le voisinage ; vérifier qu'un voisin est toujours joignable et de détecter les doublons (DAD).

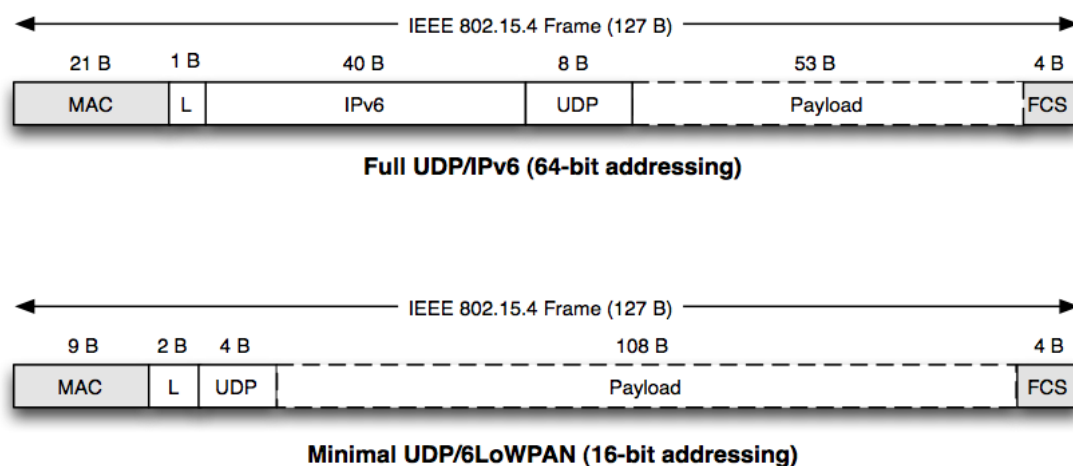


FIGURE 2.3 – Entêtes 6lowpan

Bien que proposés pour les LLNs, ce protocole est encore trop couteux [98]. Il utilise du multicast pour échanger les informations et a pour hypothèse d’avoir des routeurs et des noeuds toujours actifs et pouvant répondre à n’importe quel moment à une requête. Pour s’adapter aux contraintes des LLNs, il faut éviter le multicast, tenir compte des cycles de veilles faibles entraînant des périodes de sommeil importantes et les problématiques du multi sauts.

Le neighbor discovery optimization for low power and lossy networks [96] propose d’optimiser NDP dans le cas des LLNs. Le mécanisme de résolution d’adresses (basé sur des messages multicast entre les hôtes) est remplacé par un mécanisme de déclaration d’adresse. Ainsi, certains messages multicast associés aux configurations d’adresse des nœuds ont été remplacés par des messages unicast. Ainsi, la demande de routeur est initiée par l’hôte ce qui élimine le besoin d’annonce périodique de la part du routeur. De cette façon, les NDP pour 6LoWPAN sont plus adaptés au LLNs multi-sauts et ces mécanismes sont indépendant du protocole de routage choisis.

2.2 Protocole de routage - Routing Protocol Layer (RPL)

6LoWPAN n’offre pas de méthode de routage par défaut, or dans des cas de réseaux multi-sauts, un nœud peut être utilisé comme routeur pour faire transiter le trafic de ses voisins. Le but d’un protocole de routage est de construire et de maintenir dynamiquement les routes utilisées par les paquets pour traverser le réseau. Dans le cas d’un LLN, des compromis doivent être faits afin de maintenir un routage efficace tout en évitant de surcharger les noeuds.

Le protocole Routing Protocol Layer (RPL) construit et maintient une topologie réseau sous forme d’un graphe acyclique orienté (Directed Acyclic Graph (DAG)) ayant comme racine une ou plusieurs passerelles. Les données transmises par les nœuds du réseau ne seront transmises par les liens du DAG. RPL permet le trafic Multi-point to point (MP2P) (appelé trafic montant), Point to Multi-point (P2MP) (appelé trafic descendant) et le trafic Point to Point (P2P).

Il existe plusieurs protocoles de routage pour réseaux de capteurs, nous pouvons citer LoadNg [104] qui est un protocole réactif. Les routes y sont construites dynamiquement. Nous avons privilégié le choix de RPL [114] qui est un protocole proactif permettant de toujours disposer au niveau de la passerelle de la topologie réseau ce que nous utiliserons dans les chapitres 3 et 4.

2.2.1 Fonctionnement du protocole

RPL est un protocole de routage proactif à vecteur de distance pour LLN. Il construit un DODAG en utilisant une fonction objective et un ensemble de métriques et de contraintes. L'ensemble des paquets de signalisation utilisés pour construire cette structure sont des paquets Internet Control Message Protocol v6 (ICMPv6).

2.2.1.1 Construction du DODAG

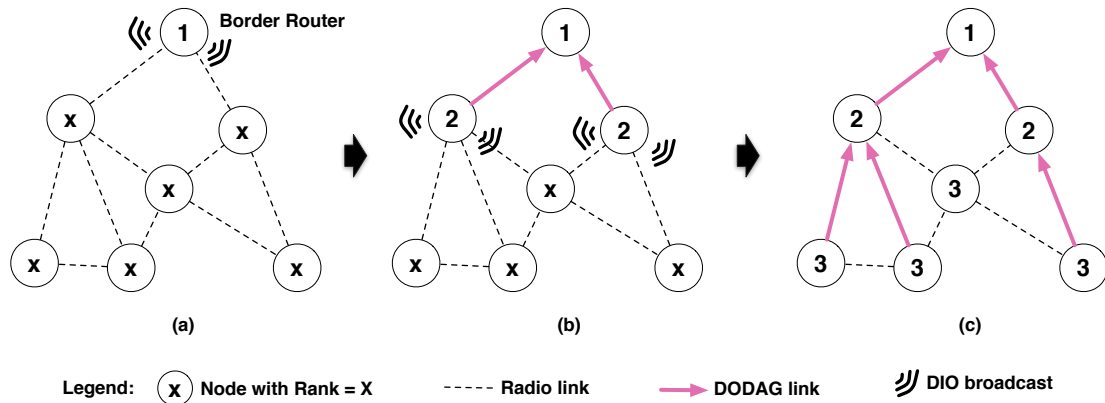


FIGURE 2.4 – Construction d'un DODAG

Le graphe construit par RPL réponds aux besoins décrits par les fonctions objectives utilisées. Le processus de construction du DODAG démarre au LoWPAN Border Router (LBR) utilisé comme racine qui est généralement le nœud collecteur de données. RPL supporte l'utilisation de racines multiples configurées dans le réseau.

Le processus est représenté dans la figure 2.4. La racine commence la diffusion des informations sur le DAG qu'elle veut construire en utilisant un DODAG Information Object (DIO). Les nœuds à portée de communication de la racine reçoivent ce DIO, et rejoignent ou non la structure selon le résultat de la fonction objective, les caractéristiques du DAG et le coût du chemin annoncé. Une fois que le nœud s'est joint à la structure, il a une route vers la racine de la structure DODAG). La racine est appelée le parent du nœud. Le nœud calcule son rang dans le graphe, qui représente la position du nœud dans la structure DODAG. Chaque nœud dans le graphe a un rang qui représente la position relative de ce nœud par rapport à la racine de la structure DODAG. La notion de rang est utilisée par RPL notamment pour éviter les boucles [114].

Si ce nœud est configuré pour agir comme un routeur dans le réseau, il commence à diffuser à son tour dans son voisinage les nouvelles informations de la structure qu'il vient de rejoindre à travers des messages DIO. Si le nœud n'est pas configuré pour être un routeur alors il rejoint tout simplement la structure DODAG et n'envoie pas de message DIO.

Les nœuds voisins recevant cette annonce vont répéter ce processus de sélection de parent, d'ajout d'itinéraire et d'annonce des nouvelles informations concernant la structure DODAG à l'aide des messages DIOs. Ce processus continue jusqu'à couvrir tous les nœuds du réseau. Chaque nœud de la structure DODAG a une entrée de routage vers son parent (ou plusieurs parents selon la fonction d'objectif) à travers lequel ce nœud peut atteindre la racine de la structure DODAG.

Les messages Destination Advertisement Object (DAO) visent à maintenir les routes descendantes et ne sont utilisés que pour des applications nécessitant des trafics de type point à multi-point et

point-à-point (dire que certains sont plus courants que d'autres). Ils sont émis par les routeurs intermédiaires dans le réseau afin que les noeuds avec un rang plus petit puisse annoncer des routes pour le trafic descendant.

2.2.1.2 Tables de routage

Afin de gérer les tables de routage, une instance RPL dispose de deux modalités : avec et sans stockage des routes.

Lorsque les routes sont stockées, une table de routage doit être construite au niveau de chaque nœud. Ceci est accompli par les message DAO. Ils sont utilisés pour annoncer les nœuds qui peuvent être des destinations potentielles dans du trafic descendant. Un nœud appartenant à la structure DODAG enverra un message DAO à ses parents. À la réception de ce message DAO, un nœud parent ajoute une entrée dans la table de routage et il envoie à son tour un message DAO à ses parents (des agrégations des informations reçues peuvent être envisagées). Ce processus se poursuit jusqu'à ce que l'information atteigne la racine du DODAG. Dans ce cas-là pour le trafic P2P le message remonte jusqu'à un nœud en commun puis le message est rerouté depuis cet ancêtre commun.

Dans le cas où les routes ne sont pas stockées, seul la racine reçoit et traite les messages DAO en provenance des nœuds et ainsi seul la racine connaît le chemin vers chaque destination. Ainsi pour atteindre un nœud, le routage est précisé explicitement par la racine. Ce cas d'utilisation peut être utile dans le cas de noeuds suffisamment contraint pour ne pas pouvoir gérer une table de routage.

2.2.2 Maintenance du DODAG

RPL est proactif et ne garantit pas un routage sans boucles ou des bornes sur les délais de convergence. Ce choix est acceptable dans des scénarios de smart metering dans lequel un délai supplémentaire occasionnel des paquets n'est pas critique [116] mais peut être dommageable dans des scénarios où des délais plus stricts sont requis.

2.2.2.1 Détection et évitement des boucles

Les boucles peuvent subvenir lors de changements de topologie réseau. Elles entraînent des pertes de paquets en raison de l'expiration du Time To Live (TTL) et des congestions. RPL peut détecter et réparer les boucles lorsque du trafic est envoyé dans le réseau et qu'une incohérence est détectée. La transmission des paquets est suspendue tant que le DODAG n'est pas remis dans un état admissible ce qui peut occasionner des délais et des surcharges des mémoires tampon. Les réparations doivent être ciblées afin de ne pas engendrer une consommation trop importante d'énergie.

Les règles utilisées par RPL pour éviter les boucles utilisent le rang précédemment introduit. Lors de la désignation d'un parent, un nœud ne peut sélectionner qu'un voisin qui a un rang plus faible que le sien. De plus, il n'est pas possible de réduire son rang pour augmenter artificiellement le nombre de parents potentiels.

Dans le cas où RPL fonctionne avec stockage, il est possible d'utiliser les entêtes pour spécifier si le trafic est montant ou descendant. Si un routeur reçoit un paquet "descendant" et qu'il doit l'envoyer à un nœud avec un rang plus faible ou un paquet "montant" et qu'il doit l'envoyer à un nœud avec un rang plus élevé alors une incohérence est détectée et une réparation locale est déclenchée.

Dans le cas où RPL fonctionne sans stockage, la racine détecte au moment de l'envoi si un routeur apparaît plus d'une fois dans le chemin emprunté.

2.2.2.2 Réparation globale et locale

RPL dispose de deux mécanismes de réparation :

Le premier mécanisme est local, lorsqu'un nœud détecte une incohérence, il se détache du DODAG en se mettant à un rang infini ce qui "empoisonne" ses routes. Les enfants le détectent et le retirent de leur liste de parents avant une reconstruction des routes.

Le second mécanisme est global. Il reconstruit l'intégralité du DODAG. La racine incrémente la version du DODAG id qu'elle propose, les nœuds propagent ce changement et reconstruisent un nouveau DODAG.

2.2.2.3 Trickle

RPL utilise un mécanisme de timer adaptatif appelé "Trickle" (goutte à goutte) afin de contrôler le débit d'émission des messages DIO. Trickle double l'intervalle séparant deux émissions successives de paquets DIO à chaque fois que le réseau est cohérent, et ce jusqu'à une valeur maximale. Ainsi le nombre de messages de contrôle diminue dans le réseau lorsqu'il est stable. Quand une incohérence est détectée, le timer est réinitialisé à sa valeur minimale. Un des principaux avantages de l'utilisation du timer Trickle est qu'il ne nécessite pas de code complexe et il est assez facile à mettre en œuvre. La convergence de Trickle vers le temps inter DIO maximal peut être difficile dans des conditions réelles de transmissions [102].

2.3 Couche applicative - Constrained Application Protocol (CoAP)

La couche applicative permet les échanges d'informations entre les différents hôtes. Il existe plusieurs protocoles applicatifs à destination des LLNs. Le groupe de travail IETF Constrained RESTful environment (CoRE) a accompli la standardisation de Constrained Application Protocol (CoAP)[97]. Ce protocole applicatif peut être vu comme une adaptation de HyperText Transfer Protocol (HTTP) pour les nœuds contraints et visant les applications M2M courantes. Le but n'est pas de compresser HTTP mais plutôt d'utiliser le même paradigme REpresentational State Transfer (REST) en plus d'offrir des fonctionnalités telles que la découverte de service ou l'envoi de messages asynchrones.

Le choix de CoAP a été motivé pour plusieurs raisons : Son paradigme REST permet d'interfacer un réseau de capteur facilement avec des services web classiques et permet d'avoir de manière pratiquement transparente un mapping entre service web et services offerts par un réseau de capteurs. Les fonctionnalités de caching sont ainsi implémentées facilement. Il ne requiert pas d'utiliser TCP et ainsi permet une gestion plus flexible d'envoi de paquets. Enfin ses mécanismes d'observations permettent de coupler différents modèles de communication PUSH afin d'avoir une gestion des envois de paquets aussi flexible que possible. En outre, ses multiples implémentations disponibles et interopérables pour systèmes contraints [57] et non-contraints [10, 58] rendaient possible la création rapide de proxy comme nous le ferons dans le chapitre 3.

2.3.1 Architecture REST & Observations

CoAP utilise l'architecture REST qui fournit une série de principes servant à construire des interfaces efficaces pour des services Web qui ont été utilisés avec succès dans HTTP [94]. Ce style préconise l'absence d'état et l'utilisation de méthodes HTTP explicites. Les ressources sont rendues disponibles via une Uniform Resource Identifier (URI) puis les clients y accèdent via des méthodes telles que : GET, PUT, POST, et DELETE. En plus des verbes classiques, CoAP dispose d'un verbe OBSERVE qui permet de créer un canal d'abonnement et ainsi d'assurer une diffusion asynchrone. Utiliser REST permet la traduction entre HTTP et CoAP. Cela aussi bien pour le proxy qui doit implémenter des traductions de protocoles que pour le développeur qui obtient une valeur d'un capteur comme il obtiendrait une valeur d'une Application Programming Interface (API) web. Enfin CoAP peut

identifier et transporter différents formats de message tels que Extensible Markup Language (XML), Javascript Serial Object Notation (JSON) ou bien Concise Binary Object Representation (CBOR) [11].

2.3.2 Modèle de message

CoAP est un protocole binaire, le modèle d'échange de message en CoAP est basé sur l'échange de messages User Datagram Protocol (UDP) avec une entête fixe de 4 octets et des options densément encodées. Chaque message contient un identifiant unique qui permet de détecter les doublons.

La sémantique des requêtes et des réponses est transportée respectivement dans les codes méthode et dans les codes réponses. Toutes les autres informations relatives à la requête comme une URI ou bien le type de message est transmis comme une option. Un jeton est utilisé pour faire la correspondance entre les requêtes et les réponses.

La fiabilité (optionnelle) est obtenue en utilisant des messages Confirmable message (CON). Un message CON est transmis avec une échéance par défaut et un temps d'attente exponentiel entre chaque tentative de ré-transmissions jusqu'à ce que le destinataire du message envoie une réponse Acknowledgement message (ACK) avec le même identifiant de message. Quand un destinataire n'est pas capable de traiter un message CON il envoie une réponse Reset message (RST) au lieu d'un ACK. Si la fiabilité n'est pas désirée comme dans le cas de mesures individuelles, il est possible d'envoyer cette réponse comme un Non-confirmable message (NON). Elles ne seront pas acquittées mais auront un identifiant afin de détecter les doublons. Si la réponse à un NON n'est pas possible, un paquet RST est envoyé en réponse.

2.3.3 Proxy et mise en cache

CoAP permet la mise en cache des réponses afin de répondre efficacement aux requêtes. Une mise en cache utilisant la fraîcheur des informations peut être prévue au niveau d'un nœud contraint ou d'un intermédiaire tel qu'un proxy. L'utilisation d'un proxy est utile dans les LLNs [25].

En premier lieu, elle permet de limiter le trafic et de ne pas surcharger les nœuds, les performances sont améliorées car de nombreuses transmissions sont évitées, les appareils contraints peuvent rester en veille. Enfin le proxy permet d'éviter de nombreuses attaques contre les nœuds.

2.4 Systèmes d'exploitation et simulation - Contiki & Cooja

Les appareils contraints ont besoin d'un système d'exploitation adapté à leurs contraintes. Nous pouvons en citer plusieurs comme : Contiki [30], RIOT [6] et OpenWSN [109]. L'utilisation de Contiki a été motivée en raison de la disponibilité d'une suite complète et intégrée d'émulateur et de simulateur directement au sein du même projet. En outre, son empreinte mémoire légère et la disponibilité d'implémentation des protocoles principaux que nous avons vus dans ce chapitre et ce dès le début de nos travaux était un point important. Il fonctionne sur les nœuds fournis par de nombreux testbeds dont IoTlab, implémente toute la pile protocolaire que nous avons vu dans ce chapitre [103, 57] et dispose de fonctionnalités de cycles de veille ainsi qu'un simulateur réseau avancé.

2.4.1 Cycles de veille - ContikiMAC

ContikiMAC [32] est le mécanisme d'endormissements par défaut de Contiki similaire à BoX-MAC [77]. Il est en charge d'éteindre et d'allumer les interfaces radio d'un nœud. Ce mécanisme de cycle de veille construit au dessus de IEEE 802.15.4 permet au nœud d'éteindre et d'allumer leur

radio périodiquement (par défaut toutes les 125 ms) pour consulter l'activité du canal dans le but d'économiser de l'énergie.

Les réveils de nœuds voisins peuvent être désynchronisés, ainsi une source devra l'envoyer à plusieurs reprises (packet *strobing*) jusqu'à la réception d'un acquittement ou bien le nombre maximal de retransmissions possibles. Lors d'un réveil, si un nœud détecte que le canal est occupé, il reste éveillé jusqu'à la réception de la trame entière. Si ce nœud est le destinataire de cette trame il reste éveillé pour le recevoir entièrement sinon il éteint sa radio. Des mécanismes complémentaires de phase-locking sont mis en place afin de réduire les envois répétés de paquets par l'expéditeur.

Ce mécanisme d'endormissement n'est généralement pas mis en place au niveau de la passerelle car il n'est pas souhaitable d'avoir des délais importants pour la racine du DODAG RPL. De plus, la passerelle est généralement un nœud non-contraint ainsi les contraintes d'économies d'énergies sont moins strictes pour elle.

2.4.2 Cooja

Cooja [81] est un simulateur réseau qui émule des nœuds avec le même binaire utilisé pour flasher un nœud réel. Cette émulation est rendue possible pour les nœuds utilisant la plateforme MSP par le logiciel mspsim [36]. Au cours d'une simulation, le simulateur modélise les interactions réseau via différents modèles de propagation et émule le comportement des nœuds. Le trafic réseau et l'exécution logicielle qui sont ainsi générés permettent l'introspection des couches basses du système. Ainsi il est possible de mettre en pause, inspecter les différents états de transmissions des nœuds, contrôler la quantité de données qu'ils émettent et reçoivent [82]. Cela rends COOJA particulièrement efficace pour étudier et construire des réseaux de capteurs et voir leur évolution sans déployer sur des nœuds réels à chaque modification.

Grace au niveau de détail offert par Contiki et Cooja, il est possible de trouver la quantité d'énergie consommée par un nœud au cours de son déploiement [33]. La quantité d'énergie consommée par un nœud est la somme de l'énergie consommée par chacune de ses parties (CPU, radio, ...). Comme COOJA permet d'avoir le détail des activités et des temps qu'un composant a passé dans chaque état, il suffit de multiplier le temps passé par la consommation nominale pour cet état.

2.5 Conclusion

Comme nous l'avons vu dans ce chapitre, les réseaux de capteurs disposent d'une pile de protocoles propre et de systèmes très différents de l'Internet usuel. Cette différence est due aux contraintes énergétiques et matérielles que ces nœuds ont. Loin d'être complètement séparée des piles protocolaires classiques, cette pile se lie à l'existante via des traductions et adaptations au niveau des passerelles. Les passerelles devront avoir les deux piles afin de remplir leurs rôle. Plus généralement, les protocoles réseaux destinés aux LLNs devront coexister et s'interopérer avec les réseaux classiques (Ethernet, Wifi, LTE, ...). Tous les appareils des LLNs n'utiliseront pas forcément ces protocoles et le domaine est suffisamment vaste pour qu'un écosystème de protocoles puisse se mettre en place. Tout au long des chapitres 3 et 4, nous présenterons nos contributions qui visent à adapter des services courants des serveurs classiques aux spécificités des réseaux de capteurs.

Chapitre 3

Cache adaptatif

There are only two hard problems in
Computer Science : cache invalidation and
naming things.

Phil Karlton

Contents

3.1 Introduction	22
3.1.1 État de l'art	23
3.1.2 Architecture & Implémentation	23
3.2 Performance d'un cache simple	24
3.2.1 Modèle théorique	25
3.2.2 Simulations & Résultats	26
3.2.3 Validation des performances	27
3.3 Performance d'un cache adaptatif	29
3.3.1 Modélisation de la satisfaction utilisateur	29
3.3.2 Analyse énergétique et modélisation du réseau	30
3.4 Optimisation multi-objectifs	34
3.4.1 Justification de la méthode d'optimisation multi-objectifs	34
3.4.2 Formalisation en problème multi-objectif	34
3.4.3 Résultats expérimentaux	35
3.5 Conclusion	37

Nous nous intéresserons dans ce chapitre à l'utilisation d'un proxy cache au niveau de la passerelle afin d'améliorer les performances d'un LLN. Les requêtes venues de l'extérieur peuvent être ralenties par les conditions difficiles de transmissions et un faible débit. De plus les problèmes de congestion et de pertes de paquets justifient d'autant plus la mise en place de solution limitant la sollicitation du LLN autant que possible. Lorsque plusieurs clients souhaitent consulter la même information, un serveur proxy permet également d'éviter les communications redondantes.

Le chapitre est organisé de la façon suivante : Nous introduisons le sujet dans la Section 3.1 en présentant l'état de l'art, l'architecture choisie et une justification de notre approche. Puis dans la

section 3.2 nous effectuons une simulation avec un cache simple pour quantifier les gains et avoir une modélisation préliminaire. C'est dans la section 3.3 que nous présentons notre contribution principale à savoir un cache qui adapte les temps de vie en fonction d'objectifs concurrents via une approche multi-objectifs qui sera détaillée dans la section 3.4. Enfin nous concluons ce volet dans la section 3.5 en présentant quelques ouvertures de nos travaux.

Bien vérifié qu'il y a une vraie précision dans l'utilisation des termes passerelle, routeur, proxy, cache. Ces mots ne sont pas synonymes

3.1 Introduction

L'utilisation de service de mise en cache afin d'améliorer la réactivité d'un système et réduire son utilisation est une pratique courante [44]. La mise en cache consiste à mettre la réponse d'une requête directement accessible au niveau de la passerelle afin de la servir comme réponse à une requête éventuelle sans solliciter de nouveau le serveur dont elle est issue. Ainsi des gains de rapidité et d'économie de la bande passante sont observés. Dans un contexte contraint, l'intérêt de ces mécanismes est encore plus justifié¹. Nous proposons dans ce chapitre d'étendre la gestion de ce proxy cache pour adapter l'utilisation et la configuration des ressources disponibles en fonction de la demande, de l'état du réseau et des informations disponibles au niveau de la passerelle.

3.1.0.1 Améliorations des performances

Une autre des principales raisons de développer une architecture de mise en cache pour les réseaux de capteurs est de permettre aux capteurs d'économiser de l'énergie par la mise en cache d'informations qui ont déjà été demandées par un client distant. De cette façon, les nœuds peuvent continuer de rester dans un état de sommeil puisque les demandes distantes sont directement servies par le proxy cache.

Cette disposition améliore également le délai de réponse à une requête entrante. En effet, la bande passante et les conditions de transmissions n'étant pas très performantes, envoyer une requête prends du temps et ce temps est inutilement dépensé s'il l'a déjà été auparavant et que la réponse alors donnée est encore valable. Lorsque la circulation est élevée, en fait, il devient plus probable qu'une valeur mise en cache peut servir à plusieurs demandes de certains clients distants. Par conséquent, l'introduction d'un système de cache permet de réduire les demandes transférées au réseau et par conséquent, de réduire considérablement paquet retransmissions ou des pertes.

3.1.0.2 Traduction facilitée d'un protocole à l'autre

Dans le cas où plusieurs protocoles sont proposés pour interagir avec le système, il peut être pertinent de traduire d'un protocole à l'autre les informations déjà obtenues. Dans le cas de certains protocoles (HTTP et CoAP), la traduction d'un protocole à l'autre est rendu facile par l'utilisation d'un paradigme commun (REST).

Ces mécanismes sont d'autant plus importants dans les cas de mécanismes d'abonnements où si rien n'est fait, un nœud devra répondre dans plusieurs protocoles à une requête redondante. Les mécanismes de mise en cache et d'observation peuvent fonctionner ensemble. Par exemple, une information déjà obtenue par observation peut être mise en cache et rendue disponible à une autre requête indépendante. Dans le cas de demandes d'abonnement émises par un client distant, le proxy les traite par le maintien d'une liste de ressources observées et une liste de clients abonnés. Chaque

1. On remarque que la consommation d'énergie en mode de veille est ordres de grandeur inférieurs à ceux de réception et d'émission des états.

fois qu'une notification de mise à jour de ressource est envoyé par le nœud vers le proxy, le proxy transmet ces messages aux abonnés correspondants par les protocoles adaptés.

3.1.0.3 Informations disponibles

Les termes de passerelles, de routeur de bordure et de reverse proxy désignent des fonctionnalités qui peuvent être offertes par un seul et même nœud. Stratégiquement placé entre l'offre et la demande, la passerelle peut contenir de multiples services tels qu'un pare-feu, de l'agrégation ou de la compression [47]. Ces services peuvent s'appuyer sur de nombreuses informations disponibles telles que la topologie du réseau, les services offerts par les nœuds, les applications extérieures qui les utilisent, les paramètres de la couche MAC, l'énergie résiduelle des nœuds. Dans les approches classiques, ces informations ne sont pas recoupées afin d'agir sur la politique de cache et la configuration du réseau. C'est l'objectif que nous allons poursuivre car des économies d'énergie sont à réaliser quand une vue globale du système est disponible.

3.1.1 État de l'art

L'IETF héberge plusieurs groupes de travail dédiés à l'étude et l'amélioration de protocoles à destination des LLNs comme nous l'avons vu dans le chapitre 2. En plus des protocoles en eux-mêmes, il existe d'autres documents en charge de spécifier des aspects techniques de la mise en place de ces protocoles [16] et en particulier des méthodes de traductions de HTTP vers CoAP. Ce corpus de spécification encourage l'interopérabilité et la définition d'interfaces claires.

Les services Web RESTful et HTTP sont largement utilisés pour publier l'état des ressources [59]. Toutefois, les services Web ne sont pas adaptés pour les LLNs en raison de la surcharge de paquets introduit par HTTP et la présence de la fenêtre de congestion TCP. Pour ces raisons, le groupe de travail IETF CoRE vise à réaliser l'architecture REST le dans une forme appropriée pour LLNs à travers la définition d'un protocole comme CoAP [97].

De nombreux travaux académiques [24, 25] s'appuient sur ces spécifications pour produire des architectures interopérables. Cependant les aspects énergétiques ne sont pas directement présents au cœur des réglages de ces proxys. L'utilisation de cache au sein même des réseaux est également une technique explorée [34], cependant les paramètres de gestion des temps de vie dans un cache sont réglés à une valeur constante et indépendante de l'état du réseau.

Notre approche consiste à gérer les paramètres de ces interfaces via des modèles et des informations disponibles au niveau de la passerelle afin de gérer au mieux ce proxy pour qu'il puisse réduire la latence d'une requête, éviter les congestions superflues dans le réseau et améliorer sa durée de vie.

TODO : Mettre des renvois aux sections dans les contributions.

Il faut clairement spécifier les contributions.

3.1.2 Architecture & Implémentation

Architecture générale Comme présenté dans la Figure 3.1, le réseau est composé d'un ensemble de nœuds connectés entre eux formant un DODAG ayant pour racine un routeur de bordure. Ce routeur traduit les paquets IPv6 classiques vers des paquets 6lowpan et les transmet dans le LLN. Une fois que la réponse revient du réseau contraint elle est traduite et renvoyée à l'expéditeur. Il est à noter que dans cette architecture, le reverse proxy est celui qui transmet la requête entrante au LLN. Ainsi dans le cas où plusieurs chemins existent pour une même destination comme c'est le cas avec des DODAG, la route la plus optimale sera choisie. Ainsi le fait de n'avoir qu'un seul cache pour plusieurs routeurs de bordure permet d'éviter les problèmes liés à des caches désynchronisés.

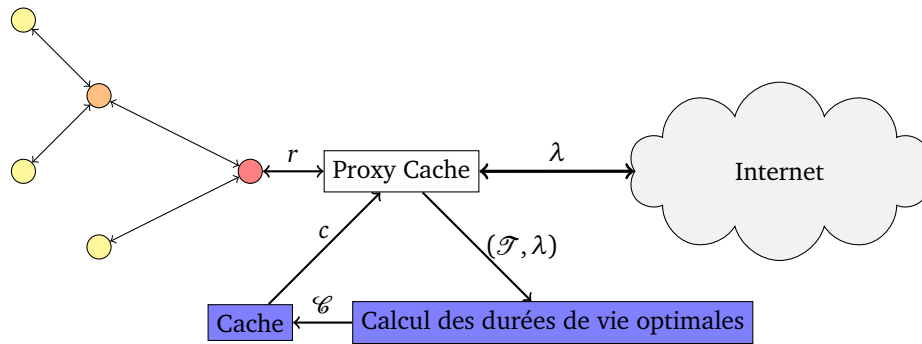


FIGURE 3.1 – Architecture du reverse proxy adaptatif

Nous sommes dans un cas simplifié dans lequel le fait d'avoir une information dans le cache nous dispense de la revalider. Cette hypothèse est essentielle car sans elle nous serions obligé de solliciter quand même le LLN.

Nous avons également une hypothèse de cache aussi grand que nécessaire. En effet, les ressources sont souvent petites et peuvent tenir facilement en RAM même pour des noeuds limités.

Ces requêtes seront ensuite envoyées au routeur de bordure du réseau de capteurs suivant une fréquence moyenne de requête r_i que l'on peut mesurer. Nous supposons par la suite que le rythme des requêtes venant de l'extérieur suit une loi de Poisson de paramètre λ_i .

Analyse Comportementale La passerelle implémente un mécanisme de cache lui permettant de garder en mémoire les résultats des requêtes passées pendant une durée de vie c_i . Une requête visant une ressource i , venant de l'extérieur, en cache depuis moins de c_i sera traitée par la passerelle sans solliciter le réseau de capteurs. Étant critiques, ces paramètres c_i doivent être choisis avec précision. Un c_i trop petit rendra la mémoire cache inefficace. À l'inverse une durée de cache trop longue risquera de donner des valeurs non pertinentes pour les applications.

La traduction de protocole est usuellement faite par le proxy. Comme nous l'avons vu dans la section 2.3, les requêtes et réponses CoAP sont envoyées en UDP et ne nécessite pas de maintien de connexions comme HTTP. Afin de réduire le délai et l'utilisation de la bande passante, les serveurs CoAP eux-mêmes peuvent mettre en cache leur réponse.

Il est à noter que les requêtes prises en compte par le cache sont les requêtes qui sont en lecture seule. Typiquement les requêtes GET. Les requêtes qui visent à modifier des ressources ne peuvent évidemment pas être mise en cache.

Choix de la configuration optimale En fonction des paramètres et des consignes rentrées par l'utilisateur, notre modèle va fournir une gamme de solution possible pour la configuration du cache. De cette population de solutions, une solution unique c va être choisie et déployée. Ce vecteur contiendra l'ensemble des temps de vie pour chaque noeud du réseau. Bien évidemment, le choix du c optimum pourra être amendée et modifié au cours du temps en fonction de l'évolution de la topologie et du réseau.

3.2 Performance d'un cache simple

Dans un premier temps, nous étudions le fonctionnement d'un cache simple. Nous pouvons d'ores et déjà voir que les intérêts immédiats du cache comprennent une réduction des délais de réponse et une supervision facilitée de notre réseau de capteur vis-à-vis des notifications et de la topologie.

3.2.1 Modèle théorique

Le taux d'arrivée de chaque requête reçu par la i ème ressource est distribué comme un processus de Poisson de paramètre λ_i (dimension : $[s^{-1}]$). Les demandes sont interceptées par le proxy qui gère également la réponse éventuelle de la donnée CoAP serveur. Si le proxy a une valeur stockée qui est assez frais, qui est dont la durée de vie est inférieure à une valeur donnée c_i (comme expliqué ci-dessous), il répond directement à la demande d'un client à distance, sans la transmettre dans le WSN. Sinon, si la valeur requise est pas présent ou il est plus âgé que c_i , il transfère la demande à l' i ème CoAP serveur. En outre, le proxy stocke les réponses des capteurs dans le cache, afin de les rendre disponibles pour d'autres demandes entrantes éventuelles.

Si la ressource demandée par le client n'est pas dans le cache ou que la donnée présente est trop vieille, le serveur va faire une requête sur la ressource en question. Soit T_i le temps moyen entre 2 requêtes arrivant sur la passerelle pour une ressource i , comme énoncé dans [68], le temps d'attente entre 2 requêtes consécutives transmises aux capteurs est de :

Le temps entre chaque requête est modélisé par un temps d'attente exponentiel de paramètre λ_i . Dans la mesure où les requêtes sont aléatoires et uniformément réparties entre les différents nœuds on démontre [68] que le temps d'arrivée entre deux requêtes est une loi exponentielle de paramètres $T = NT_i = \frac{N}{\lambda_i}$.

On peut définir MR (cache Miss Ratio) comme la probabilité qu'une requête ne puisse pas être satisfaite par les informations disponibles dans le cache. Ainsi, MR correspond à la probabilité que le temps entre chaque arrivée d'une requête soit plus grande que c_i :

$$MR = \int_{c_i}^{\infty} \frac{e^{-\frac{t}{T}}}{T} dt = e^{-\frac{c_i}{T}}. \quad (3.1)$$

Ainsi on déduit h (Cache Hit) qui est le complémentaire de MR : $h = 1 - e^{-\frac{c_i}{T}}$.

Expliquer ici que l'on a affaire à un ratio.

Combien de temps me faut-il pour me rendre compte qu'un nœud est indisponible ? Dans ce genre de cas il faudra au plus le temps c_i plus le temps de transmission.

Est-ce que ça fonctionne avec des topologies aléatoires ? Changeante (Ajout ou perte de nœuds) ? Tant que la passerelle a connaissance de l'ajout d'un nœud, elle peut recalculer son modèle à la volée. Dans le cas où un nœud est jugé mort, alors elle peut le retirer de sa liste également et recalculer les temps de caches optimaux en fonction de la nouvelle topologie.

Le mécanisme de mise en cache peut être résumé de la façon suivante : Lorsqu'un client envoie une requête à destination d'un nœud du LLN, Dans le cas où le proxy ne dispose pas de la réponse à la requête demandée, elle est transmise au nœud concerné. Une réponse est récupérée (on récupère un code d'erreur dans tous les cas). Une fois que la passerelle a récupéré la valeur de réponse, elle l'a met dans son cache pour un temps donné. Ce temps est appelé le temps de vie de la ressource dans le cache. Ainsi si une nouvelle requête arrive pendant ce temps de vie, le proxy pourra y répondre sans solliciter les nœuds.

Le temps d'inter arrivées T_i (dimension : $[s]$) entre deux requêtes consécutives peut être modélisé comme un variable aléatoire exponentielle de paramètre λ_i . Puisque dans nos simulation la destination est choisie aléatoirement, en moyenne, on attends N requêtes consécutives avant d'avoir une nouvelle requête sur le même nœud i . Ainsi, avec l'hypothèse que λ_i est le même pour tous les N nœud dans le réseau, le temps d'inter arrivées est distribué avec une loi exponentielle de paramètre $T = NT_i = N/\lambda_i$.

(TODO : Mettre une justification avec un scénario de smart building dans lequel tous les thermostat sont à peu près équivalent)

Cache Miss ratio (m) peut être défini comme la probabilité qu'une requête ne soit pas servie par le proxy. Ainsi, m correspond à la probabilité que l'inter arrivées entre deux requêtes vers un même nœud soit plus grande que c_i , c'est à dire 3.1

Ainsi, h peut être exprimé comme le complément de m :

$$h = 1 - e^{-\frac{c_i}{T}} \quad (3.2)$$

Comme montré dans le schéma de la figure 3.1, les clients finaux envoient une requête de demande d'observation à l'un des N CoAP choisi au hasard. Le rythme des arrivées est distribuée selon une distribution de Poisson de paramètre λ_i , où i désigne le nœud sur laquelle la requête est envoyée. Comme expliqué précédemment, pour chaque observation il est possible de définir une durée de vie en cache c_i , qui désigne combien de temps la réponse à une requête est valide avant d'être retirée par le proxy et une nouvelle requête doit être envoyée au serveur CoAP.

Dans nos simulation, nous avons pour hypothèse que chaque nœud n'a qu'une ressource CoAP disponible. En outre, les valeurs de c_i admettent des bornes c_{min} et c_{max} .

Si la ressource requise par le client distant se trouve pas dans la mémoire cache ou le valeur mise en cache est dépassée, le proxy émet une requête CoAP au donné CoAP serveur. Étant donné que la demande pour le i ème nœud arrive, sur moyenne, chaque T_i , il peut être prouvé que la durée moyenne r_i (dimension : [s]) entre deux demandes consécutives à un même nœud peuvent être défini comme :

$$r_i = \begin{cases} \lceil \frac{c_i}{T_i} \rceil T_i & \text{si } T_i \leq c_i \\ T_i & \text{sinon.} \end{cases} \quad (3.3)$$

3.2.2 Simulations & Résultats

Afin d'évaluer le modèle de serveur cache introduit précédemment, nous effectuons la simulation suivante. 12 serveurs CoAP utilisant Contiki comme système d'exploitation sont déployés et émulsés via Cooja. Nous utilisons le nombre de requêtes arrivant sur le LLN comme métrique. Afin d'avoir une empreinte mémoire faible, chaque serveur n'a qu'une seule ressource et afin de faciliter l'analyse sera un texte de taille fixe et maximale ne causant pas de fragmentations. La figure 3.1 représente la topologie utilisée.

Notre implémentation de proxy est basée sur la bibliothèque libre Californium [59]. La raison principale de ce choix est que Californium mettait déjà en œuvre un ensemble de fonctionnalités de base en CoAP. Nous avons ensuite introduit deux mécanismes de stockage d'informations. Le premier est la base de données de mise en cache, où toutes les informations de la Wireless Sensor Networks (WSN) sont réunis et faire des provisions et pour laquelle nous avons choisi CouchDB, car il offre une interface REST en HTTP pour interroger les résultats enregistrés. C'est à notre connaissance, le premier travail présentant un proxy HTTP/CoAP mis en œuvre pour évaluer l'impact la consommation énergétique.

Dans la figure 3.2, nous montrons un schéma du système de cache que nous avons implémenté où N serveurs CoAP exposent une ressource.

Selon le schéma de la figure 3.1, un client choisit d'abord au hasard l'un des CoAP serveurs, puis envoie une requête GET afin de connaître l'état de la ressource sélectionnée. Le temps entre deux requêtes consécutives est distribué selon une distribution exponentielle de paramètre λ .² Depuis la requête de CoAP ressources sont équiprobables, en exploitant la superposition fondé des processus de Poisson, on peut dire que le taux de l'arrivée demandes de chaque nœud est encore une distribution

2. On remarque que λ est le taux d'arrivée des demandes au proxy.

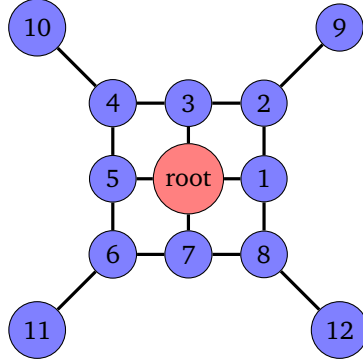


FIGURE 3.2 – La topologie radio considérée. $N = 12$ noeud placés sur une grille avec la racine comme noeud central.

de Poisson de paramètre $\lambda_i = \lambda/N$, où i désigne la ressource sur le i ème noeud, donc le taux des demandes pour un générique i ème noeud d'arrivée est un processus de Poisson qui a pour paramètre λ_i .

Les demandes du client distant sont interceptés par le proxy, qui traduit les requêtes HTTP en CoAP et, inversement, traduit les réponses CoAP en réponses HTTP. En outre, le proxy met en cache les réponses rendues par les noeuds afin de les mettre à disposition pour une éventuelle autre demande entrante. La fraîcheur de la cache valeur pour le i ème noeud est notée c_i . Si une demande de statut de i ème arrive ressource, le proxy vérifie d'abord dans le cache si elle a déjà une valeur disponible pour être envoyé au client, qui est une valeur dont la durée de vie est inférieure à c_i , sinon il transmet la demande. En outre, le proxy magasins la valeur recueillies dans le cache pour les futures demandes et transmettre le recueillis des informations au client final. Comme dans tout système de mise en cache, rafraîchissement paramètres sont très critiques. Si c_i est trop grande, l'information ne sera pas être suffisamment renouvelé et les valeurs mises en cache peuvent souvent être différente de la valeurs réelles. Si c_i est trop petit, alors le proxy sera sous-performant, le gaspillage d'énergie et de ne pas atteindre la performance optimale.

Les simulations ont été effectuées générer 50 requêtes client pour les serveurs CoAP dans le LLN. Pour liée possible les fluctuations statistiques dans les résultats de la simulation, les résultats de simulation sont obtenu en faisant la moyenne sur 10 essais consécutifs.

Les paramètres principaux de la modélisation du système sont listés dans la table 3.1.

Les paramètres de consommation d'énergie présentés dans ce tableau ont été prises à partir de la fiche produit interne d'un noeud prototype. Comme les autres noeuds de capteurs commerciaux bien connus utilisant le chipcon de CC2420 (par exemple, Arbalète MicaZ, Berkeley Telosb [88]), la consommation d'énergie est plus élevé dans le mode de réception que dans le mode de transmission à pleine puissance.

Afin de simplifier les simulations, nous avons considéré les mêmes valeurs de λ_i , c_i , et le même cycle de service pour chaque noeud CoAP dans le WSN.

3.2.3 Validation des performances

Afin d'évaluer les performances de notre implémentation, nous évaluons d'abord le cache hit ratio h comme une fonction de la durée de vie c_i . Nous indiquons pour chaque courbe l'intervalle de confiance à 2σ , où σ est la déviation standard sur plusieurs exécutions successives.

La simulation (traits pleins) et théoriques (lignes en pointillés) sont représentées pour différentes

Débit de transmission	R	250 kbps
Intervalle entre deux tentatives de préambules	T_p	0.4 ms
Temps pour détecter un paquet ACK	T_d	0.16 ms
Taille d'une requête (GET)	L_r	87 octets
Taille d'un paquet de réponse	L_a	96 bytes
Temps pour transmettre un IEEE 802.15.4 ACK	$T_{\mathcal{A}}$	0.608 ms
Puissance consommée lors de la transmission	$P_{\mathcal{T}}$	0.0511 W
Puissance consommée lors de la réception	$P_{\mathcal{R}}$	0.0588 W
Puissance consommée lors du sommeil	$P_{\mathcal{S}}$	$2.4 \cdot 10^{-7}$ W
Nombre de nœuds dans le LLN	N	12

TABLE 3.1 – Paramètres utilisés dans les simulations.

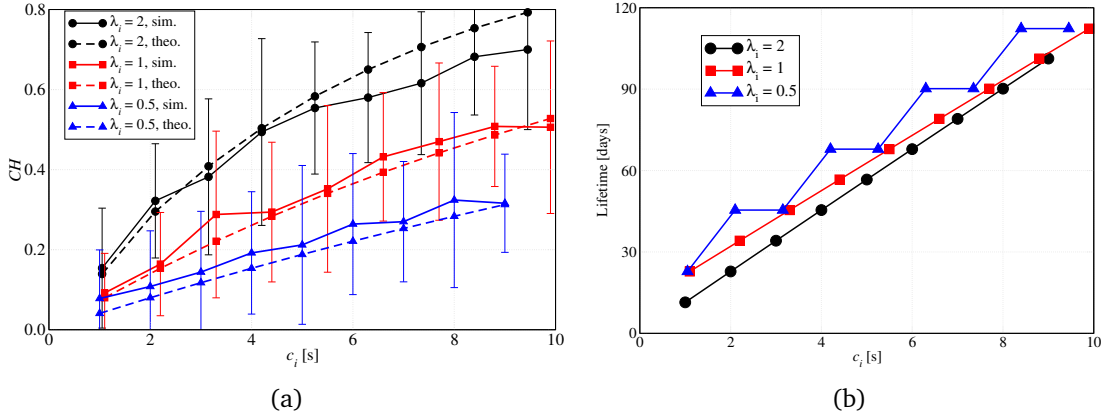


FIGURE 3.3 – (a) h comme fonction de la durée de vie de cache c_i . Simulation (traits plein) et théorique (pointillés). (b) Durée de vie du LLN comme fonction de la durée de vie dans le cache c_i . $\lambda_i = 2$ (cercles), $\lambda_i = 1$ (carrés), et $\lambda_i = 0.5$ (triangles).

valeurs de paramètres λ_i . Bien entendu, plus la valeur de λ_i augmente, plus le nombre de demandes par unité de temps augmente aussi. Toutes les courbes de la figure 3.3 (a) croissent quand le paramètre durée de cache augmente. Il est clair que plus la mémoire cache durée est élevée, plus la probabilité que la requête est servie par le serveur proxy, ne pas être transmis aux WSN nœuds. De même, pour une valeur donnée de durée du cache est élevé, plus le nombre d'arrivée par unité de temps est élevé, plus la probabilité que le proxy répond au client avec une valeur mise en cache. La effet bénéfique de l'utilisation de la mise en cache ne se limite pas à l'énergie économisées liées à la diminution du nombre de paquets transférés à la WSN avec l'utilisation de proxy.

L'utilisation d'un système de mise en cache permet également d'économiser l'énergie et étendre la durée de vie du réseau, définie comme le temps nécessaire pour avoir au moins un nœud à court d'énergie. Dans la figure 3.3 (b) nous pouvons voir l'impact de la mise en cache sur la durée de vie de notre système. Intuitivement, un c_i entraîne une plus grande durée de vie du réseau car un plus grand nombre de demandes sont servies par le cache sans être transféré au réseau. La forme en escalier de la courbe avec $\lambda_i = 0.5$ est due à la définition de r_i dans (3.3), différentes valeurs de c_i conduisent au même nombre de requêtes transmises à la WSN.

Nous considérons que la durée de vie du réseau est *le temps qui sépare son démarrage du premier arrêt définitif d'au moins un des nœuds*, que tous les nœuds démarrent avec la même énergie initiale. Le réseau que nous utilisons est composé de 12 nœuds fixes connectés via un DODAG. Des requêtes sont générées et envoyées vers la passerelle qui décide de celles à traiter en utilisant le cache. Les nœuds économisent de l'énergie en utilisant ContikiMAC en tant que protocole d'accès à la couche de liaison. Ce protocole introduit des cycles de veille pour allumer périodiquement l'interface radio. On suppose que les transmissions sont effectives et sans erreurs. On suppose que la simulation commence lorsque le DODAG de RPL a convergé.

Dans la Figure 3.3 nous pouvons voir que plus λ_i est élevé, plus le cache est intéressant. Il l'est d'autant plus que les constantes de temps c_i sont élevées, car les probabilités d'accéder au cache augmentent et ainsi le nombre de requêtes transférées vers le réseau de capteurs diminue. Il est à noter que dans l'hypothèse d'un rythme de requête élevé, il est particulièrement intéressant de mettre en cache. En raison des collisions de paquets et des situations de trafic élevé, l'énergie consommée lors de ces situations est considérable. Ainsi en plus des requêtes qui sont économisées on gagne également sur les coûts liés aux collisions de paquets.

3.3 Performance d'un cache adaptatif

Afin d'avoir une utilisation plus efficace des ressources, nous allons élaborer une technique utilisant à la fois les volontés des utilisateurs mais aussi les objectifs de l'application à rester en fonctionnement pour une certaine durée de vie. L'optimisation a deux buts, le premier est de fournir une configuration de durée de cache admissible pour les exigences de satisfaction de l'utilisateur. Le second but, réciproque du premier, est d'estimer quelle peut-être la satisfaction de l'utilisateur pour une durée de vie du réseau imposée.

Il est nécessaire de définir certains paramètres pour caractériser la stratégie d'optimisation présentée. Tout d'abord, nous définissons la durée de vie d'un réseau comme l'intervalle de temps entre son démarrage et la perte de son premier nœud à court d'énergie.

Le choix de la durée de vie optimale en cache dépend de l'application. Selon l'énergie d'un nœud, nous pouvons décider d'augmenter la durée de vie de la valeurs mises en cache, en acceptant des valeurs inférieures à jour, mais, de l'autre côté, étendant la durée de vie du réseau, en raison de la réduction du nombre de demandes transféré à la WSN.

Afin de trouver des solutions optimales, nous prenons pour hypothèse que la topologie du réseau est statique pendant le temps de traitement de la requête, que les pertes de paquets sont négligeables car le réseau est en hypothèse de trafic faible.

Nous étendrons les fonctionnalités de ce cache en montrant qu'il peut être utilisé afin d'optimiser la consommation énergétique des nœuds. En utilisant le modèle analytique de ContikiMAC, et la topologie de routage, nous construisons une stratégie d'optimisation des temps de vie des caches qui permet d'atteindre une durée de vie maximale pour une satisfaction donnée ou bien d'avoir la satisfaction maximale pour une durée de vie donnée.

Nous verrons aussi comment maximiser les deux en utilisant des solutions optimales non dominées de Pareto.

3.3.1 Modélisation de la satisfaction utilisateur

Il est nécessaire d'introduire également une métrique de satisfaction de l'utilisateur. Pour notre modélisation nous allons considérer que plus une information donnée à l'utilisateur est récente, plus ce dernier est satisfait. Cependant, à rythme de requête égal, une durée de vie dans le cache courte signifie plus de requêtes pour le LLN et donc un coût énergétique plus important. Ces valeurs exogènes

au problème dépendent fortement de l'application en question. Une valeur haute de c_i sera utilisée pour des nœuds ayant peu d'énergie ou avec des informations peu changeante. Par contre, une valeur c_i petite sera pour des informations devant être aussi récentes que possible.

Dans notre approche, nous introduisons la satisfaction d'un utilisateur comme une métrique à maximiser. Plus la valeur de c_i est courte plus l'utilisateur est satisfait. Comme présenté dans la Figure 3.3(a), quand la durée de temps de vie est courte, il est probable que les requêtes entrantes seront envoyées avec le LLN ce qui entraînera une consommation d'énergie et une réduction de la durée de vie. D'un autre côté, la maximisation de la durée de vie requiert de garder des ressources aussi longtemps que possible dans le cache afin de réduire sa consommation.

Nous introduisons les quantités c_{min} et c_{max} représentant les bornes admissibles d'une valeur c_i . Il est inutile de demander une information plus récente que c_{min} et une ressource plus vieille que c_{max} n'est plus pertinente. Nous pouvons voir que les meilleures durées de vie possible sont obtenues quand $c_i = c_{max}$ mais cela se fait généralement au détriment de la satisfaction.

Il faut aussi indiquer que nous ignorons tous les effets de protocoles de routage. On ne tient compte que du trafic applicatif.

Ainsi nous introduisons γ_i mesurant la satisfaction d'un utilisateur :

$$\forall i, \gamma_i = \frac{c_{max}(i) - c_i}{c_{max}(i) - c_{min}(i)} \quad (3.4)$$

En particulier, une satisfaction est minimale quand $c_i = c_{max}$ et une satisfaction est maximale quand $c_i = c_{min}$.

3.3.2 Analyse énergétique et modélisation du réseau

Il faut expliquer à quoi rime cette section. Autrement c'est juste des maths dont on ne voit pas facilement l'intérêt.

Nous ne prenons en charge que le calcul du trafic applicatif.

Il faut également vérifier que les unités sont cohérentes.

Les nœuds utilisent ContikiMAC comme protocole d'accès à la couche liaison afin d'éviter les collisions de transmissions et contrôler l'accès au médium. L'utilisation de l'interface radio étant le principal consommateur de la batterie dans nos simulations, nous avons analysé ce protocole afin d'estimer la durée de vie du réseau.

Il y a 4 états possibles pour un nœud utilisant ContikiMAC : (i) transmission, (ii) réception, (iii) mode sommeil et (iv) écoute du canal avec les consommations énergétiques suivantes : $\Omega_{\mathcal{T}}$, $\Omega_{\mathcal{R}}$, $\Omega_{\mathcal{S}}$, et $\Omega_{\mathcal{L}}$ (dimension : [W]), respectivement.

$$\Omega = \Omega_{\mathcal{S}} + \Omega_{\mathcal{L}} + \Omega_{\mathcal{T}} + \Omega_{\mathcal{R}} \quad (3.5)$$

où : $\Omega_{\mathcal{S}}$ est l'énergie consommée pendant la période de sommeil du i -ème nœud ; $\Omega_{\mathcal{L}}$ est l'énergie demandée pour écouter le canal sur une période fixée ; $\Omega_{\mathcal{R}}$ est l'énergie utilisée pour la réception d'un paquet ; $\Omega_{\mathcal{T}}$ est l'énergie utilisée pour transmettre un paquet.

Nous ne pouvons pas agir efficacement sur les termes $\Omega_{\mathcal{S}}$ et $\Omega_{\mathcal{L}}$ car ils dépendent fortement du protocole MAC utilisé. C'est sur $\Omega_{\mathcal{T}}$ et $\Omega_{\mathcal{R}}$ que nous allons agir via notre cache. Une modélisation plus précise peut-être trouvée dans [68].

Justification de l'algorithme Il est également nécessaire de prendre en compte la topologie du réseau puisque les messages relayés par un nœud affectent tous les nœuds dont il dépend pour la transition de ses paquets. Le modèle devenant complexe dans le cas général, nous avons donc développé un calcul approché de la consommation énergétique qui intègre la topologie du réseau.

3.3.2.1 Énergie résiduelle

Nous pouvons dire que la consommation énergétique de chaque nœud est donnée par la somme des consommations de ses composants. Pour simplifier, seul les modules de communications seront inclus dans notre modèle énergétique. L'énergie résiduelle d'un nœud i à un instant t peut être exprimé par :

$$E_r(t) = E_0 - \Omega t \quad (3.6)$$

E_0 est l'énergie initiale du nœud considéré et Ω (dimension : [W]) est la puissance consommée par le système. Pour simplifier, nous considérons que tous les nœuds possèdent la même énergie initiale. Selon la description du protocole Contiki MAC [32], il y a 4 états possibles pour un nœud : (i) transmission, (ii) réception, (iii) sommeil et (iv) l'écoute. Ces états correspondent à des puissances consommées notées $\Omega_{\mathcal{T}}$, $\Omega_{\mathcal{R}}$, $\Omega_{\mathcal{S}}$, et $\Omega_{\mathcal{A}}$ (dimension : [W]), respectivement.

Nous définissons le temps T_C comme la durée entre deux phases actives consécutives et $T_{\mathcal{S}}$ comme la durée de la phase de sommeil du nœud considéré.³ L' Ω de l'équation (3.6) peut être exprimé par l'équation 3.5

3.3.2.2 Temps de transmission & réception

Il existe 3 types de nœuds dans notre modèle : (i) Les serveurs CoAP, (ii) les routeurs qui agissent aussi comme serveurs CoAP et (iii) une racine du DODAG. Puisque la taille d'une requête et d'une réponse ont des tailles différentes, nous distinguerons le temps pour transmettre une requête : $T_r = L_r/R$ de celui de la réponse défini comme $T_a = \frac{L_a}{R}$, où L_r et L_a sont les tailles de paquet de la requête et de la réponse respectivement et R est le débit de transmission des nœuds. D'après le protocole Contiki MAC [32] quand un nœud transmet un paquet, il reste durant $T_{p,\mathcal{T} \rightarrow \mathcal{T}}$ en transmission et pour une période $T_{p,\mathcal{T} \rightarrow \mathcal{R}}$ en réception afin de recevoir le paquet ACK du destinataire. Ces périodes de temps peuvent être exprimées par :

$$S_{p,\mathcal{T} \rightarrow \mathcal{T}} = \frac{3 + \lfloor \frac{T_{\mathcal{S}} - S_p}{S_p} \rfloor}{2} S_p \quad (3.7)$$

$$S_{p,\mathcal{T} \rightarrow \mathcal{R}} = \frac{3 + \lfloor \frac{T_{\mathcal{S}} - S_p}{S_p} \rfloor}{2} T_d + S_{\mathcal{A}} \quad (3.8)$$

où T_p indique un paquet générique qu'il soit une requête T_r ou une réponse T_a . T_d est le temps requis pour détecter avec succès un acquittement d'un récepteur, et $T_{\mathcal{A}}$ est le temps nécessaire pour transmettre un ACK. L'équation 3.7 est obtenue en faisant la moyenne entre le meilleur cas de transmission (c'est-à-dire quand le nœud commence à transmettre alors que le destinataire vient de se réveiller), et le pire cas (Le destinataire vient de rentrer en sommeil alors que le nœud commence à transmettre). De la même façon, quand un nœud reçoit un paquet, il passe une partie de son temps en réception et une partie de son temps en transmission puisqu'il a besoin de transmettre un ACK à l'expéditeur du message. Ces périodes de temps peuvent être exprimées par :

$$T_{p,\mathcal{R} \rightarrow \mathcal{R}} = \frac{3T_p}{2} + T_p \quad (3.9)$$

$$T_{p,\mathcal{R} \rightarrow \mathcal{T}} = T_{\mathcal{A}} \quad (3.10)$$

3. La durée de la phase active peut être évaluée comme : $T_{A_i} = T_{C_i} - T_{\mathcal{S}_i}$.

où T_p est l'intervalle entre chaque transmission de paquet $T_{\mathcal{A}}$ est la durée de transmission d'un paquet ACK. L'équation (3.9) est obtenue en faisant la moyenne entre le meilleur et le pire des cas. Le meilleur étant quand le paquet n'a besoin d'être transmis qu'une fois et le pire qui est celui où le nœud se réveille juste après le début d'une transmission par l'expéditeur. Dans ce cas, le nœud doit attendre la prochaine transmission pour tenter de le recevoir correctement.

Les termes $\Omega_{\mathcal{T}}$ et $\Omega_{\mathcal{R}}$ vont être différents selon le nœud visé. Nous distinguons trois cas, celui d'un serveur CoAP simple ($\Omega_{server,\mathcal{T}}$ et $\Omega_{server,\mathcal{R}}$), d'un routeur ($\Omega_{router,\mathcal{T}}$ et $\Omega_{router,\mathcal{R}}$) ou bien de la racine du DODAG ($\Omega_{root,\mathcal{T}}$ et $\Omega_{root,\mathcal{R}}$).

3.3.2.3 Consommation des serveurs applicatifs simples

Dans le cas d'un serveur applicatif CoAP qui ne fait que recevoir une requête et transmet une valeur observée, la puissance consommée durant la transmission et la réception peut alors être exprimée par :

$$\Omega_{server,\mathcal{T}} = \frac{P_{\mathcal{T}} T_{a,\mathcal{T} \rightarrow \mathcal{T}} + P_{\mathcal{R}} T_{a,\mathcal{T} \rightarrow \mathcal{R}}}{r_i} \quad (3.11)$$

$$\Omega_{server,\mathcal{R}} = \frac{P_{\mathcal{R}} T_{r,\mathcal{R} \rightarrow \mathcal{R}} + P_{\mathcal{T}} T_{r,\mathcal{R} \rightarrow \mathcal{T}}}{r_i} \quad (3.12)$$

en replaçant les tailles de paquets dans les requêtes et leurs réponses respectives dans les équations (3.7), (3.8), (3.9), et (3.10). Les termes $P_{\mathcal{T}}$ et $P_{\mathcal{R}}$ désignent la puissance consommée par un nœud en phase de réception et de transmission.

3.3.2.4 Consommation de la racine du DODAG

Dans le cas d'une racine, la puissance consommée pour transmettre à un nœud générique i et recevoir sa réponse peut être exprimée par :

$$\Omega_{root,\mathcal{T}} = \frac{P_{\mathcal{T}} T_{r,\mathcal{T} \rightarrow \mathcal{T}} + P_{\mathcal{R}} T_{r,\mathcal{T} \rightarrow \mathcal{R}}}{r_i} \quad (3.13)$$

$$\Omega_{root,\mathcal{R}} = \frac{P_{\mathcal{R}} T_{a,\mathcal{R} \rightarrow \mathcal{R}} + P_{\mathcal{T}} T_{a,\mathcal{R} \rightarrow \mathcal{T}}}{r_i} \quad (3.14)$$

Puisque la racine transmet à chacun de ces enfants, la puissance consommée pour transmettre à tous les nœuds et recevoir peut être exprimée par :

A inspecter

$$\Omega_{root,\mathcal{T}} = \sum_{i=1}^N \Omega_{root,\mathcal{T}} \quad (3.15)$$

$$\Omega_{root,\mathcal{R}} = \sum_{i=1}^N \Omega_{root,\mathcal{R}} \quad (3.16)$$

3.3.2.5 Consommation des nœuds relais/serveurs

Les routeurs répondent aux requêtes qui les concernent mais servent aussi d'intermédiaires celles à destination de leurs enfants. Ainsi la puissance consommée pour transmettre les paquets peut être exprimée par :

$$\Omega_{router,\mathcal{T}} = \sum_{j \in m_i} (\Omega_{server,\mathcal{T}_j} + \Omega_{root-T_j}) + \Omega_{server,\mathcal{T}} \quad (3.17)$$

$$\Omega_{router,\mathcal{R}} = \sum_{j \in m_i} (\Omega_{server,\mathcal{R}_j} + \Omega_{root-R_j}) + \Omega_{server,\mathcal{R}} \quad (3.18)$$

où m_i désigne l'ensemble des enfants du nœud i comme montré dans la figure 3.1. Les termes à la droite de (3.17) et (3.18) sont introduits car ils peuvent aussi répondre eux-mêmes à des requêtes applicatives.

3.3.2.6 Consommation en phase d'écoute de canal & sommeil

Enfin, la puissance consommée en écoute de canal et dans l'état de sommeil peut être exprimée par :

$$\Omega_{\mathcal{L}} = \frac{T_{fl} P_{\mathcal{R}}}{T_C} \quad (3.19)$$

$$\Omega_{\mathcal{S}} = \frac{T_{\mathcal{S}} P_{\mathcal{S}}}{T_C} - \Gamma_{\mathcal{T}} - \Gamma_{\mathcal{R}} \quad (3.20)$$

où $P_{\mathcal{S}}$ est la puissance durant l'état de sommeil, $\Gamma_{\mathcal{T}}$ et $\Gamma_{\mathcal{R}}$ sont deux termes correctifs. En temps normal, un nœud effectue soit une écoute du canal, soit transmet ou reçoit un paquet soit dort. $\Gamma_{\mathcal{T}}$ et $\Gamma_{\mathcal{R}}$ sont utilisés pour raffiner la puissance consommée durant la phase de sommeil. Les périodes de sommeils chevauchent celles de transmissions ou réceptions sur des temps courts ainsi sans ces termes la puissance consommée sera surestimée. $\Gamma_{\mathcal{T}}$ et $\Gamma_{\mathcal{R}}$ peuvent être exprimée par :

$$\Gamma_{\mathcal{T}} = \Omega_{\mathcal{S}} \frac{1}{T_C} \left(\frac{3 + \lfloor \frac{T_{\mathcal{S}} - T_p}{T_p} \rfloor}{2} (T_p + T_d) + T_{\mathcal{A}} \right) \quad (3.21)$$

$$\Gamma_{\mathcal{R}} = \Omega_{\mathcal{S}} \frac{1}{T_C} \left(\frac{3T_p}{2} + T_p + T_{\mathcal{A}} \right) \quad (3.22)$$

Le terme $\Gamma_{\mathcal{T}}$ tient du fait que durant les opérations de transmissions comme les phases de (i) transmissions stroboscopiques d'un paquet sur une phase $T_{\mathcal{S}}$, (ii) la transmission standard d'un paquet et (iii) la réception d'un acquittement, un nœud serait normalement en sommeil. Ainsi le facteur correctif $\Gamma_{\mathcal{T}}$ est nécessaire puisque autrement l'énergie consommée par un nœud avec ce modèle serait surestimée car réception et transmission se chevaucheraient avec les opérations de sommeil normales sur une période. Des considérations similaires peuvent être dressées pour le terme $\Gamma_{\mathcal{R}}$. Quand un nœud attends un acquittement pour transmettre un paquet ACK, pour recevoir le préambule et le paquet, le nœud serait normalement en état de sommeil.

En injectant les expressions (3.21) et (3.22) dans (3.20) et les expressions (3.11), (3.12) (si il s'agit d'un nœud CoAP autrement (3.17) et (3.18) pour un routeur ou (3.15) et (3.16) pour la racine), (3.19), et (3.20) dans (3.5), il est possible de dériver une expression pour la consommation qui ne dépends que de la topologie et des paramètres de communications.

3.4 Optimisation multi-objectifs

Le modèle analytique que nous avons introduit peut-être utilisé pour trouver une valeur adéquate de c_i qui permet de remplir si possible la consigne de durée de vie minimale du réseau. Comme nous l'avons vu dans INSERER REF CORRECTE, la meilleure durée de vie peut être atteint avec c_i . Cependant, si on veut maximiser la satisfaction d'un utilisateur γ , définie comme $\gamma = \sum_{i=1}^N \gamma_i / N$, alors $c_i = c_{min}$ ce qui a pour effet de minimiser la durée de vie.

Les algorithmes génétiques sont connus et utilisés dans les réseaux de capteurs sans fils pour avoir un placement optimal des noeuds [38, 52]. Cependant l'utilisation de méthodes génétiques pour trouver les paramètres optimaux d'un cache n'ont pas encore été explorés.

3.4.1 Justification de la méthode d'optimisation multi-objectifs

Les méthodes d'optimisation multi-objectifs sont nombreuses. La plus courante consiste à affecter des poids sur chaque fonction que l'on cherche à maximiser puis de les sommer afin de se ramener à un problème avec un seul objectif. Cette approche nécessite une détermination des poids qui n'apparaît que comme arbitraire et nous n'avions pas de moyens clairs de déterminer quel paramètres étaient les plus importants. Cependant dans un cas où la priorité de chaque nœud est définie précisément, cette méthode serait intéressante.

Les méthodes génétiques à l'inverse d'autres méthodes plus sophistiquées ne nécessitent pas des calculs complexes ou de bibliothèques logicielles spécialisées.

L'approche que nous avons choisie en algorithme génétique se justifie aussi par le fait qu'il est souvent désirable d'avoir plusieurs solutions à un même problème et de pouvoir changer facilement de l'un à l'autre en fonction des situations. A l'inverse des méta-heuristiques qui cherchent une solution unique, les algorithmes génétiques créent une population de solutions présentant différents compromis.

C'est une méthode versatile. Les modèles de durées de vie et de satisfaction peuvent être changés. Ce qui compte c'est l'approche.

3.4.2 Formalisation en problème multi-objectif

TODO : Expliquer ici l'algorithme.

Chaque solution étudiées de notre problème va être encodée sous la forme d'un vecteur contenant les durées de cache pour les nœuds correspondants. À chaque itération (appelée génération) l'algorithme génétique va croiser des individus, affecter des mutations et sélectionner un certain nombre d'individus pour aller à l'étape suivante.

3.4.2.1 Mutation

La mutation est un processus de transformation qui va se jouer avec une chance p_m sur chaque individu d'une génération à l'autre. Nous sommes restés avec le processus de mutation polynomial par défaut utilisé dans NSGAI et détaillé dans [29]. Ce processus produit une solution mutante dont chaque gène est dans les bornes inférieures et supérieures admissibles. Il utilise un paramètre η_m dont la valeur est usuellement laissé à 20.

3.4.2.2 Croisement (Crossover)

Le croisement est un mécanisme d'exploration des solutions possibles. C'est un opérateur d'algorithme génétique utilisé pour mélanger les gènes de deux solutions pour en former deux autres.

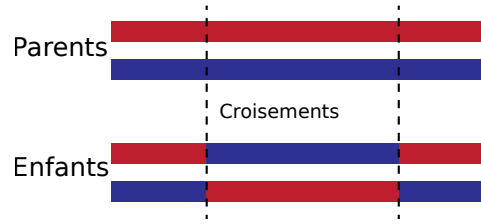


FIGURE 3.4 – Mécanisme de crossover

3.4.2.3 Aptitude (Fitness)

La fonction d'aptitude est celle qui se charge d'évaluer à quel point une solution est admissible ou non et si elle l'est quelle est son efficacité. Déterminer une fonction d'aptitude est la partie la plus liée au problème lors de la conception d'un algorithme génétique. Les fonctions de sélections, mutations et de croisement sont assez génériques d'un problème à l'autre.

$$f(c, g, \lambda) = (\gamma, \mathcal{L}) \quad (3.23)$$

Il faut préciser ici que l'on va prendre la moyenne des satisfactions car on a quand même besoin d'avoir une grandeur scalaire à augmenter.

3.4.2.4 Sélection

Le processus de sélection choisit quels individus survivent d'une génération à l'autre. Les méthodes de sélections sont variées cependant nous avons retenues celle présentée dans NSGA2. C'est une méthode élitiste, les meilleures solutions sont retenues d'une génération à l'autre et forment les élites. Cependant cette sélection s'accompagne d'une prime à la diversité. A niveau d'aptitudes semblables les solutions les plus éloignées les unes des autres sont sélectionnées. Cette diversité dans les solutions est un point important. Il permet de peupler le front de Pareto sur une large portion.

3.4.3 Résultats expérimentaux

Le choix de valeur adéquate pour c_i introduit un compromis entre la durée de vie du réseau et la satisfaction des utilisateurs. En outre, comme certains nœuds agissent comme routeurs, ils réduisent le nombre de transmissions pour le i -ème nœud affectant la durée de vie du nœud i mais aussi la durée de vie des routeurs qui sont sur le chemin entre le nœud i et la racine. Afin de résoudre ces problèmes multiobjectifs, nous utilisons Non-dominated Sorting Genetic Algorithm (NSGA)II [28]. Si on fournit, la durée de vie du réseau visée et les grandeurs c_{min} et c_{max} ainsi que la topologie du réseau, une solution si elle existe est donnée et le jeu de paramètres c_i optimaux qui satisfont les contraintes de durée de vie tout en maximisant la satisfaction des utilisateurs γ .

Expérimentalement nous définissons $c_{min} = 1$ s et $c_{max} = 9$ s, $\lambda_i = 1$ s⁻¹, et une topology radio avec 12 nœuds comme montré dans la Figure 3.1. Nous avons mis les contraintes sur la durée de vie du réseau à respectivement 25, 50, et 75 jours, et nous avons évalué l'ensemble c^* qui maximise la satisfaction des utilisateurs tout en remplissant la durée de vie du réseau. A titre de comparaison nous montrons les résultats obtenus avec $c_i = c_{min}$ et $c_i = c_{max}$. Les résultats sont montrés dans la table 3.3.

Population	P	100
Probabilité d'accouplement	p_c	0.5
Probabilité de mutation	p_m	0.2
Nombre de génération	n_g	50
Index de la distribution de mutation	η_m	20

TABLE 3.2 – Paramètres utilisés dans l'optimisation multi-objectifs.

c_i	Durée de vie [jours]	γ
c_{min}	11.4219	100%
c_{25}^*	25	95%
c_{50}^*	50	68%
c_{75}^*	75	49%
c_{max}	101.256	0%

TABLE 3.3 – Résultats des performances de différentes stratégies dans les cas où $c_i = c_{min}$ et $c_i = c_{max}$. Pour estimer c^* , avec des durées de vie de 25, 50, et 75 jours.

$$\begin{array}{ll} \text{maximize} & \gamma = \frac{1}{N} \sum_{i=1}^N \gamma_i, \end{array} \quad (3.24)$$

$$\begin{array}{ll} \text{maximize} & \forall i, c_i \end{array} \quad (3.25)$$

$$\begin{array}{ll} \text{subject to} & c_{i,min} \leq c_i \leq c_{i,max}, \end{array} \quad (3.26)$$

$$(3.27)$$

L'ensemble des c_i obtenus grâce à l'outil d'optimisation permet de répondre de manière efficace à l'exigence sur la durée de vie du réseau, tout en maximisant la satisfaction utilisateur.

Si aucune contrainte sur la durée de vie sont imposées, trouver un point de travail adapté (ie, un ensemble de valeurs de c_i approprié) pour le système peut être considéré comme un problème d'optimisation multi-objectif. Dans ce cas, l'algorithme NSGA II permet de retrouver le front non-dominée de Pareto des solutions, qui est l'ensemble des valeurs qui sont à l'optimum de Pareto. Une solution est Pareto optimale quand il est impossible d'améliorer un objectif sans réduire au moins l'un des autres objectifs. Dans Figure 3.5, le front de Pareto pour la mise en cache présenté l'architecture est représentée.

Afin de trouver un compromis acceptable entre durée de vie du réseau et satisfaction utilisateur, nous utilisons une méthode d'optimisation multiobjectif NSGA II [28] combinée à une modélisation exposée en détail dans [68].

Comme présenté dans la Figure 3.5, nous obtenons pour chaque durée de vie admissible la meilleure satisfaction utilisateur γ_i possible [68]. Nous obtenons ainsi un front de Pareto permettant de choisir parmi les solutions optimales celle qui est adaptée à notre application en fonction des informations provenant des différentes couches logiques.

La raison pour laquelle nous obtenons un front de Pareto en forme de droite affine provient de nos hypothèses de départ. En effet, le rythme d'arrivée des requêtes est supposé uniforme, et tous les nœuds sont identiques. Dans ces conditions, nous obtenons une loi linéaire sur les défauts de cache

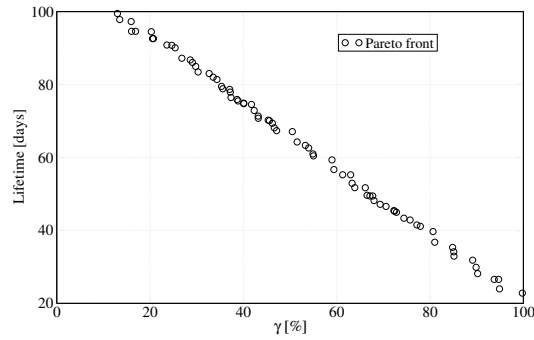


FIGURE 3.5 – Front de Pareto pour le scénario envisagé.

et la consommation du réseau qui en dépend. Cependant, dans le cas où la couche MAC pourrait être dynamiquement configurée en fonction des informations disponibles, la relation serait beaucoup plus complexe et la modélisation linéaire impossible.

Selon les résultats précédents, il est possible de trouver un ensemble de solutions appropriées qui permettent d'obtenir la meilleure configuration des durées de vie de cache. Le solveur associés à chacun de ces points un ensemble de paramètres c_i qui peuvent être utilisés sur le réseau start-up à régler correctement le la durée de cache.

Nous rappelons que la stratégie d'optimisation présenté ci-dessus a été évaluée selon une définition spécifique de Quality of Service (QoS). Cependant, elle peut être facilement généralisée afin d'englober d'autres fonctions objectives et contraintes dans le processus d'optimisation.

La fonction économique que l'on va utiliser peut être modifié selon différents critères. Par exemple on peut avoir une répartition des poids différentes selon qu'un nœud a plus ou moins de voisins. Une autre approche consiste à prendre en compte la répartition des popularités des requêtes envoyées.

TODO : Quel est le délai pour détecter un gros problème dans le réseau et adapter le niveau de requêtes histoire de calmer le jeu ? Notre mécanisme et notre définition de satisfaction est insensible aux conditions réseaux comme une charge temporairement élevée. On peut imaginer une autre définition de la satisfaction qui stipule que l'on équilibre la charge en fonction des facultés de chaque nœuds. Notre modèle de résolution multi-objectifs s'adaptera a cette nouvelle contrainte.

Il y a de très nombreux paramètres (les durées de vie de cache, les bornes la satisfaction qui peut prendre des formes très diverses et non-linéaires) Fixer une contrainte de durée de vie ne donne pas trivialement une solution de configuration de cache. Si on a une fourche par exemple on peut avoir plusieurs solutions qui aboutissent a la même durée de vie sur le réseau. D'où l'intérêt d'avoir une variété de solutions larges afin d'offrir de nombreuses opportunités.

3.5 Conclusion

Ce chapitre a présenté une stratégie pour améliorer la qualité de service d'un réseau de capteurs en adaptant dynamiquement les paramètres de la mémoire cache en fonction d'informations en provenance de différentes couches logiques du système. En premier lieu, nous avons vu qu'un cache standard aidait à économiser significativement de l'énergie puisqu'il empêche les communications redondantes triviales. Puis nous avons introduit un modèle d'optimisation exploitant les différents critères des utilisateurs ainsi que les informations extraites du cache. Ce qui nous a permis de configurer de manière optimale la gestion de la durée de vie des informations au sein du cache.

Ce chapitre a abordé le problème de l'énergie QoS efficaces optimisation utilisant des techniques

entre couches et exploitant une spécifiquement introduites plate-forme de mise en cache. Nous avons d'abord présenté la mise en œuvre d'une mise en cache solution basée sur un nœud proxy qui est en charge de répondre, si un cache valeur est disponible, à une demande provenant d'un client distant sans transférer à la WSN. Les résultats de simulation montrent que l'introduction d'un l'architecture de mise en cache a un impact positif en termes d'économie d'énergie sur le système le rendement, car il permet de réduire les transmissions à l'intérieur du WSN. Alors, nous avons introduit une méthode d'optimisation qui, exploitant les informations recueillies par le RPL protocole et donné un ensemble de contraintes sur le minimum et les valeurs maximales de la durée de cache, permet de configurer de manière optimale les valeurs des durées de vie de mise en cache. La stratégie d'optimisation proposée permet soit trouver des solutions adaptées à la présence de contraintes sur la durée de vie du réseau ou à savoir l'ensemble non-dominée optimale de solutions dans le cas d'optimisation multi- objectifs.

Une ouverture possible de ces travaux serait d'exploitation de nos simulations sur une plateforme de plus grande envergure, telle qu'IoTlab [40], permettrait d'obtenir des informations supplémentaires sur la consommation énergétique réelle des capteurs.

Une autre ouverture serait d'étendre à un nombre quelconque de ressources concurrentes, chacune ayant des popularités concurrentes et une satisfaction sur chaque ressource différente.

Chapitre 4

Supervision active & passive

There are more things in heaven and earth,
Horatio, than are dreamt of in your
philosophy

Shakespeare - Hamlet (1.5.167-8)

Contents

4.1 Introduction	41
4.1.1 Justification	42
4.1.2 État de l'art	42
4.1.3 Contribution	43
4.2 Modélisation	44
4.2.1 Consommation énergétique de transmission et réception	45
4.2.2 Strobbling en CSMA/CA	46
4.3 Supervision passive	47
4.3.1 Supervision passive du trafic réseau	48
4.3.2 Résultats expérimentaux - Topologie en chaine	48
4.3.3 Supervision active & passive	51
4.4 Conclusion	53

Montrer que mon systeme rends le systeme plus efficace pour les questions de monitoring et d'accès

L'énergie est classiquement considéré comme une ressource critique dans les LLN Ces réseaux sont composés de minuscules dispositifs qui Auto- organisent autour d'un ou quelques-uns em passerelles, qui peut avoir différents rôles de simple référence ou le trafic coule au orchestrateur de réseau complet.

Cette passerelle pourrait influencer sur le comportement du réseau, par exemple en diminuant de l'activité quand l'énergie devient rare. C'est ce que nous avons vu dans le chapitre 3. Il faut cependant être en mesure de superviser les nœuds et leur énergie consommée afin de déduire l'énergie restantes pour chacun des nœuds. En effet, cette passerelle est sur le trajet de tout le trafic entrant dans le LLN . Cet échantillon de trafic peut être utilisé pour acquérir une estimation de la consommation

individuelle d'énergie des nœuds. L'exactitude de cette estimation peut alors être améliorée par une signalisation explicite si nécessaire.

Nous évaluons, par simulation, la précision de l'estimateur quand la topologie est connue et dans celle où elle ne l'est pas. Notre contexte est IEEE 802.15.4 avec RPL et ContikiMAC. Nous observons que la connaissance de la topologie n'est pas suffisante et nous le corrigeons via des requêtes explicites afin de prendre en compte les multiples tentatives de transmissions, les collisions de paquets et la contention.

Grâce à des simulations, nous évaluons les différentes stratégies et montrons que la la connaissance de la topologie du réseau améliore considérablement estimations. Utilisation supervision active périodique, nous démontrons également comment la passerelle peut en apprendre davantage sur la consommation résiduelle induite par le protocole de routage et la radio conditions (affirmation, collisions, effet stroboscopique).

La supervision d'un LLN est essentielle pour s'assurer qu'il est provisionner correctement et fonctionne de manière nominale. Afin de prévoir des capteurs réserve d'énergie et d'adapter le comportement du réseau en conséquence, il est nécessaire de avoir une perception précise de la décharge des batteries. Toutefois, cette informations peut ne pas être toujours disponible et sa collection à la passerelle, où les politiques mondiales sont décidées, introduit des dépenses considérables. Ainsi, mesures d'énergie indirecte en utilisant un nombre limité de messages de commande peuvent être utile de fournir une estimation précise de l'énergie résiduelle.

La supervision d'un réseau et en particulier sa consommation d'énergie est un aspect critique dans les LLNs. Afin de prédire efficacement le comportement du réseau, il est nécessaire d'avoir une vue sur la consommation énergétique et l'utilisation des ressource réseau. Cependant cette information n'est pas toujours disponible ou possible et même lorsqu'elle l'est, son cout peut être prohibitif au vue des contraintes des LLNs. Ainsi, des méthodes de supervision indirectes utilisant un nombre faibles de messages de controles peut être utile pour fournir une estimation de la consommation énergétique des noeuds.

La consommation d'énergie est un aspect crucial pour les réseaux de capteurs sans fil (WSN). Afin de prédire de manière efficace et d'adapter le comportement du réseau, il est nécessaire de avoir une perception précise de la décharge des batteries. Toutefois, cette informations peut ne pas être toujours disponibles et il peut introduire considérable les frais généraux. Ainsi, des mesures d'énergie indirectes en utilisant un nombre limité de contrôle les messages peuvent être utiles pour fournir une estimation précise de l'énergie résiduelle.

Dans ce chapitre, nous présentons un estimateur de trafic et de consommation énergétique utilisant la topologie et le trafic observé à la passerelle pour prédire de la consommation d'énergie des nœuds. En analysant les paquets atteignent la passerelle, ils prévoient la consommation d'énergie des nœuds qui (a) envoyés, (b) transmis, et (c) ont reçu ces paquets. Ils estiment également de l'énergie la consommation en raison de protocoles de routage. Les informations sur le trafic en réseau est utilisée pour modéliser restante durée de vie de chaque nœud et superviser le fonctionnement du réseau.

Nous appliquons cet estimateur sur les réseaux IEEE 802.15.4 via des nœuds basés sur Contiki émulsés dans COOJA. Afin de minimiser la consommation d'énergie au niveau des dispositifs, nous utilisons des cycles de veilles et un routage de faible puissance protocole tel que RPL. Les nœuds génèrent périodiquement le trafic vers la racine de RPL. Nous comparons nos estimations par rapport aux valeurs de consommation d'énergie réelle recueillie par le simulateur.

Les résultats montrent que les messages de supervisions sont nécessaires pour avoir une vision précise du réseau avec la pile protocolaire que nous utilisons. Il est également possible avec une hypothèse de stabilité du trafic de fournir des estimations du temps de vie restant dans le réseau.

Les résultats soulignent que RPL messages de contrôle ont un impact élevé, en particulier au début de la simulation. Nous constatons également que le meilleur estimation est obtenue en tenant compte de l'acheminement et de la radio topologie. Enfin, les simulations ont montré que les protocoles

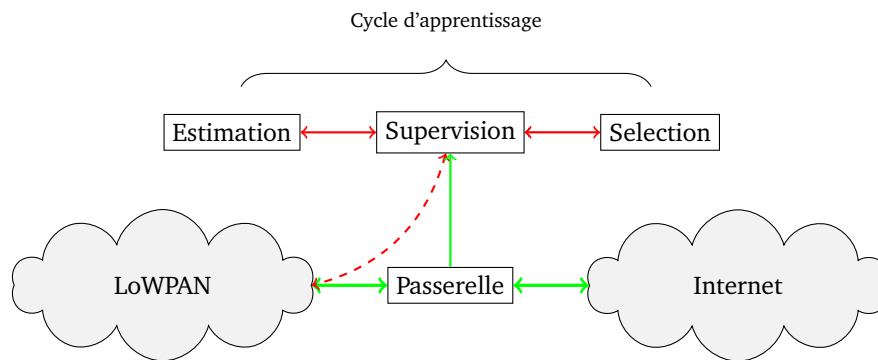


FIGURE 4.1 – Supervision et sélection de nœuds

d'application impact sur la consommation d'énergie est facilement prévisible.

Nous introduirons dans la section 4.1 les concepts principaux utilisés dans le chapitre et la justification générale de notre approche en comparaison avec l'état de l'art ainsi que notre contribution. Les modèles seront exposés dans la section 4.2 et testé dans la section ?? . La section ?? présente nos estimateurs et la section ?? les évalue. Finalement la section 4.4 conclura ce chapitre et proposera des ouvertures à ces travaux.

Questions ouvertes

- Est ce que l'on peut avoir des phénomènes d'oscillation du système ? Une estimation du durée de vie qui n'est pas bonne, entraîne une série de mécanisme de qualité de service qui amène une estimation à être différente puis se rendre compte que la configuration d'avant était meilleure et fournir des estimations fausses encore ? En gros des phénomènes de larsen.
- est ce que ce système dans le cas ou il se plante bloque des utilisateurs comme un ascenseur ou bien c'est plutôt un escalier de tel sorte qu'il ne bloque pas les utilisateurs d'en trouver un autre si il se plante.

4.1 Introduction

TODO : Parler avec des termes issus du machine learning. fit, predict, transforme...

La passerelle est souvent vue comme la source et la destination d'une grande partie du trafic applicatif. Elle joue un rôle de référence dans plusieurs protocoles comme RPL [114] ou bien des couches MAC comme IEEE 802.15.4 e [83]. Plus récemment, la passerelle a été considérée comme un l'organisateur principal du réseau qui gère les routes et organise l'accès au médium comme dans le cas de 6TiSCH [1] à l'IETF. La passerelle est dans de nombreux déploiement le nœud qui a la connaissance la plus détaillée du réseau : elle relaie le trafic applicatif entrant et sortant du LLN, elle joue un rôle central dans plusieurs protocoles de routage notamment dans RPL. Ainsi elle peut acquérir une bonne vue du réseau en terme de (i) topologie, (ii) ressources offertes, (iii) allocation des ressources radio, et (iv) l'état de fonctionnement des nœuds. Avec cet ensemble de connaissance, la passerelle peut fournir une estimation sur de multiples métriques par exemple le temps de vie restant.

L'estimation de la consommation énergétique est importante pour les systèmes embarqués. La connaissance du temps de vie restant d'un appareil d'un LLN peut avoir des conséquences importantes sur une application déployée et son dimensionnement. Il est souvent difficile d'avoir une estimation correcte d'un réseau déployé. Les raisons peuvent être multiples : l'accès à l'information de batterie restante peut ne pas être disponible. Dans le cas où elle l'est, une solution naive au problème de supervision consiste à mettre une supervision active et périodique. Même dans le cas où ce type de

solution est possible, elle est consommatrice d'énergie et ne prends pas du tout en compte le contexte de cycle de sommeil des nœuds pour fonctionner avec pour conséquence des délais importants et des congestions.

4.1.1 Justification

Il faut dire que la supervision active n'est pas toujours possible. Les nœuds peuvent ne pas avoir de fonctionnalité de réponse à des mécanismes de supervisions. Mais on veut quand même avoir une idée de l'utilisation de ce nœud.

Il faut orienter notre contribution sur cette angle là aussi.

Tu veux pouvoir réduire la facture de supervision.

On produit un certain nombre de résultats gratuitement. L'idée c'est que si tu veux certains types d'informations que tu peux pas trouver par supervision passive, tu payes avec une supervision active.

Quel est l'impact réel de l'inférence ?

Quel place pour la déduction ?

Comment mesurer la performance d'une méthode de déduction ?

4.1.2 État de l'art

Quelques contributions ont examiné le problème général de la surveillance d'un réseau de capteurs efficace en énergie. Par exemple, [70] et [62] considèrent le problème de la sélection d'un sous-ensemble de capteurs "sondeurs" chargés de surveiller activement les autres capteurs "sondés". Les sondeurs sont en charge d'émettre des alarmes vers la passerelle si ils détectent une anomalie. [70] propose un algorithme distribué d'approximation pour sélectionner un nombre minimum de sondeurs et étudie le taux de faux positif généré. [62] propose de réduire la dépense énergétique en utilisant des paquets de contrôle de routage pour sélectionner les sondeurs et en intégrant les rapports de suivi dans les messages de contrôle du protocole de routage. Ces approches nécessitent des informations explicites acquies de nœuds de capteurs, pendant que nous attendons que cette information soit facultative. **TODO : On va finir par demander explicitement des informations au nœuds il faut revoir notre position par rapport a l'état de l'art**

Dans [19], les auteurs proposent LiveNet, une architecture de surveillance semi-passive qui repose sur les sondes situés dans le réseau. En utilisant les traces agrégées transmises à la passerelle, LiveNet est capable de reconstruire topologie de réseau et de déterminer divers paramètres de performance du réseau. Ce travail vise explicitement surveillance de l'énergie, mais pourrait être adaptée à d'autres indicateurs de performance. Cependant, il nécessite la transmission et le traitement de traces dans le réseau, il est donc pas entièrement pertinent pour notre objectif.

Dans [118], les auteurs introduisent une méthode distribué pour créer un carte de l'énergie d'un LLN. Les nœuds déclarent leur niveau d'énergie résiduelle à un voisin nœud, en charge de l'agrégation et la compression de ces informations et ne transmettent que des mises à jour incrémentielles (condensés) à la passerelle. Suivant cette idée, [75] laisse l'estimation et la prédiction de temps de vie à chaque nœud puis se charge de l'envoyer au moniteur de réseau. De plus, [75] compare une méthode probabiliste, basée sur les chaînes de Markov, et une méthode statistique, sur la base d'un modèle auto-régressif, avec un simple, méthode de déclaration explicite. Dans [49], les auteurs étendent cette idée en modélisant l'énergie de chaque nœud avec un modèle de Markov caché dont les coefficients sont l'écoute avec des mesures explicites. Dans [18], les auteurs construisent une carte de l'énergie et changent la structure de surveillance régulièrement pour redistribuer le coût de cette surveillance de façon équitable à travers le réseau. Si l'idée de construire une carte de l'énergie du réseau est étroitement liée à notre premier but, toutes les méthodes mentionnées ci-dessus reposent fortement sur les rapports de l'énergie explicite et continus des nœuds. Ces rapports ne peuvent pas

être totalement évités, comme le montre nos expériences. Cependant, nous croyons qu'une légère partie de l'information peut être directement extraite de paquets de contrôle existants, sans supplé-mentaires les frais généraux.

De même une contribution peut être faite sur comment acquérir ces informations de manière intelligente.

TODO : Est ce qu'il y a des approches avec du piggybacking ? Pour estimer les conditions de trafic réseau sans doute pas.

4.1.3 Contribution

Nous proposons d'étudier la précision des mécanismes de supervision passive. Elle se propose d'estimer la consommation énergétique en utilisant les informations de trafic réseau capturées au niveau de la passerelle.

Estimer l'énergie consommée par les nœuds est une tâche intéressante pour obtenir une estimation de la durée de vie du réseau. Par exemple quand l'énergie devient trop faible un ordre peut être donné afin de faire dormir les nœuds plus longtemps. Cette dégradation de performance peut être décidé par la passerelle si les circonstances s'y prête. D'autres actions possible pourrait être de filtrer les acquisitions de données superflues, de changer le flux de trafic en influençant le protocole de routage...

Nous montrerons une approche passive et verrons que les informations de topologie dans un cas de topologie multi-sauts est une information importante. L'exploitation naïve de la source et de la destination dans les paquets capturés ne suffit pas à produire une estimation précise. Nous verrons que la connaissance de la topologie améliore les performances.

Puisque la supervision fonctionne depuis la passerelle. Elle ne saisi pas la consommation énergétique induire par le contrôle de trafic ou par les mécanismes de la couche MAC. Pour cette raison, nous introduisons des mécanismes de supervision active qui peuvent corriger nos estimations et adapter nos biais d'observations.

TODO : Mettre des renvois aux sections dans les contributions.

Nous introduisons des mécanismes de supervision passive qui sont un moyen transparent d'estimer la consommation énergétique des nœuds d'un LLN dans une topologie multi-sauts. Cette supervision passive fonctionne à la passerelle et observe au niveau de la couche réseau les sources et destination de chaque paquet. Tous les transmissions et réception de paquets consomme de l'énergie, la radio est l'un des premiers consommateurs de batteries. Un grand volume de transmission impliquera une grande quantité d'énergie consommée. Nous utilisons le lien entre transmission et énergie consommée pour offrir une estimation de l'énergie consommée par un nœud. En déduisant du trafic passant à la passerelle nous pouvons créer un modèle du trafic réseau.

Nous prenons également en compte les consommations engendrées par le relayage des paquets dans une topologie multi-sauts en utilisant les informations de topologie disponibles à la passerelle. Ces informations sur la topologie sont combinées avec une modélisation des protocoles pour estimer le temps passé par un nœud à transmettre et recevoir des messages de la part de ses voisins. Si l'information sur la quantité d'énergie qu'ils ont en réserve est disponible alors il est possible d'avoir une estimation du temps de vie qu'il reste pour chaque nœuds.

Il faut ajouter une justification que transmission et consommation sont liés.

- Estimer la quantité d'énergie consommée afin de prédire quand leur réserve d'énergie s'épuisera. (Toujours utile dans le cas où le nœud ne peut fournir ce type d'informations. On a pas toujours des possibilités de supervision avancée.)
- Réduire la facture énergétique de la supervision du réseau.
- Cette approche est indépendante de la définition de la durée de vie utilisée. On ne travaille que sur l'énergie consommée.

-
- Les informations utilisées pour nos estimations sont de toute façon utile pour superviser les performances brutes du réseau (Délai, Bande passante et charge)
 - La structure des routes est d'ores et déjà présente avec le protocole de routage RPL. Ainsi on ne fait qu'utiliser ce qui est déjà présent. De même pour le trafic qui allait de toute façon passer par la passerelle.

4.2 Modélisation

Faire plein de références au chapitre modèle et ne rappeler ici que l'essentiel

Nous utilisons des nœuds utilisant le modèle que nous avons établi dans le chapitre 2.

Dans la plupart des LLN, l'interface radio est la principale consommatrice d'énergie. Le micro contrôleur et le reste du système opère à des fréquences basses et seul les opérations d'écriture de la mémoire flash ont besoin d'une énergie comparable. En particulier, l'énergie consommée par la transmission et la réception sont en générale similaire. En effet, alors qu'envoyer des données nécessite de créer un signal radio, la réception des trames de données nécessite l'activation de systèmes électroniques de filtrage pour trouver le signal du bruit noise [51]. Dans le reste du chapitre, nous ne considérons que la consommation de l'interface radio.

On peut parler de l'overhearing comme un phénomène qui rajoute du bruit et de la perturbation dans la supervision des nœuds

TODO : Préciser le type de technologies, est ce que c'est purement en CSMA/CA. En TDMA ça paraît peu probable si on a un ordonnanceur malin. Est ce que ça change quelque chose avec les beacons au préalable ?

Dans plusieurs technologies et implémentation, quand un source émet une trame, tous les nœuds qui sont à portée radio et qui sont éveillés vont tenter de décoder la trame et devront la saisir complètement avant de vérifier qu'elle est correcte par sa somme de contrôle et qu'elle est destinée au nœud en question. Il est possible qu'un voisin éteigne sa radio lorsqu'il détecte qu'il n'est pas concerné par cette transmission.

Néanmoins, chaque nœud réveillé dépense toujours une quantité minimale d'énergie, pour analyser une trame qui est émise par l'un de ses voisins. Un cycle de veille efficace peut mitiger cet effet en gérant les réveils et endormissement. Cependant l'ordonnancement du réseau de transmissions peut être très difficile lorsque le profil de trafic est pas prévisible.

Si nous regardons un nœud i et $N(i)$, ses voisins qui sont à portée de transmission. Où i transmet une trame f de $\mathcal{L}(f)$ octets à un autre nœud $j \in N(i)$ tous les nœuds non-endormis dans $N(i)$ consomme de l'énergie. Nous considérons que les nœuds qui appartiennent $N(i) \setminus \{j\}$ et ceux qui ne sont pas endormis peuvent limiter la réception et le traitement des trames qui ne leur sont pas destinées d'une taille $\mathcal{U}(f)$ octets avant qu'elle n'éteigne leur interface radio. L'énergie dépensée pour la transmission de cette trame est :

$$\begin{cases} \text{Emitter (node } i) & : S_{cost}(\mathcal{L}(f)) \\ \text{Receiver (node } j) & : R_{cost}(\mathcal{L}(f)) \\ \text{Awake over-hearers}(N(i) \setminus \{j\}) & : R_{cost}(\mathcal{U}(f)) \end{cases}$$

où $\gamma \in [0; 1]$ modélise la fraction de trames qui seront entendues durant le cycle de veille. En utilisant les mêmes notations, nous pouvons aussi exprimé le coût d'envoi d'une trame depuis un nœud i vers un nœud j pour l'émetteur, et le récepteur

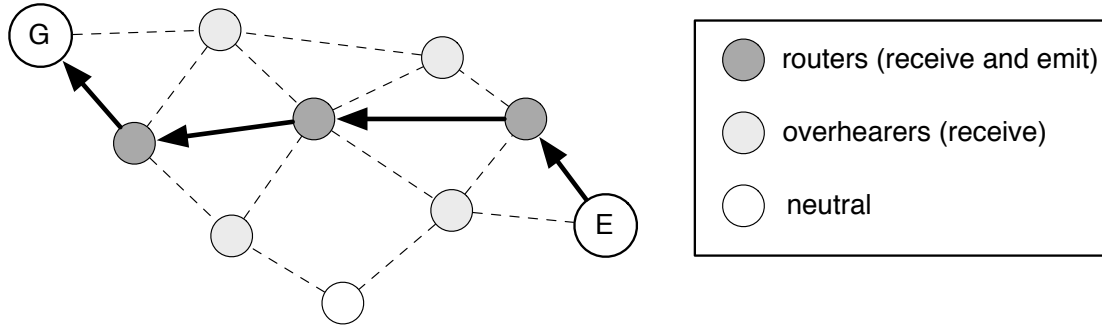


FIGURE 4.2 – Nœuds impactés par l’acheminement d’une trame depuis le nœud E vers le nœud G.

$$E_i(t) = \sum_{j \in N(i)} \left(\sum_{f \in \mathcal{F}_{i,j}(t)} \underbrace{S_{cost}(\mathcal{L}(f))}_{\text{Emissions}} + \sum_{f \in \mathcal{F}_{j,i}(t)} \underbrace{R_{cost}(\mathcal{L}(f))}_{\text{Receptions}} + \sum_{k \in N(j) \setminus \{i\}} \sum_{f \in \mathcal{F}_{j,k}(t)} \underbrace{\gamma \cdot R_{cost}(\mathcal{O}(f))}_{\text{Overhearing}} \right),$$

L’accès au canal dans IEEE 802.15.4 rends $\mathcal{L}(f)$ comme une fonction linéaire de la taille de trame et dans la plupart des implémentations $\mathcal{U}(f) = \mathcal{L}(f)$.

Nous verrons dans la section 4.2.1 utilise ce modèle pour dériver le cout de chaque transmission.

Parce que nous connaissons pour une plateforme fixée la consommation énergétique de ces états nous pouvons calculer l’énergie dépensée par un nœud en sachant le temps qu’il a passé dans un état précis.

Nous obtenons dans la section 4.2.1 une mesure de l’énergie dépensée.

4.2.1 Consommation énergétique de transmission et réception

Nous devons calculer combien d’énergie est dépensé par chaque nœud pour transmettre une trame m de taille $s(m)$ octets envoyés en unicast et avec un acquittement.

IEEE 802.15.4 fournit le temps requis pour transmettre le message m de la manière suivante :

$$T_p(m) = \left(\frac{8s(m)}{R} + \left\lceil \frac{s(m)}{L} \right\rceil h \right)$$

où $s(m)$ est la taille d’une trame IEEE 802.15.4 (exprimée en octets) contenant m , $R = 250$ kbit/s est le débit du IEEE 802.15.4, L est la charge utile (payload) maximale d’une trame IEEE 802.15.4 (127 octets) et h est le temps requis pour transmettre l’entête d’une trame. Le standard IEEE 802.15.4 fournit $h = 992\mu s$ pour les entêtes ce qui est confirmé en simulation. Puisque les entêtes sont envoyées pour toutes les trames nous prenons aussi en compte la surcharge causée dans le cas d’une fragmentation de paquets.

Soit P_{TX} et P_{RX} les puissance requises pour emettre et recevoir des données respectivement. Si on multiplie $T_p(m)$ par P_{TX} (respectivement P_{RX}) nous obtenons l’énergie dépensée pour transmettre (respectivement recevoir) m . Les paramètres peuvent être trouvés dans les notices d’utilisations des composants utilisés. Par exemple, la radio Chipcon CC2420 [21] implémentant IEEE 802.15.4 opère avec une tension de $V_{DD} = 3V$, le courant pendant la réception est de $I_{RX} = 19.7 mA$ et le courant durant une émission à 0 dBm est $I_{TX} = 17.4 mA$ [26].

Nous pouvons donc estimer $S_{cost}(m)$ et $R_{cost}(m)$, l’énergie nécessaire pour respectivement envoyer et recevoir une trame :

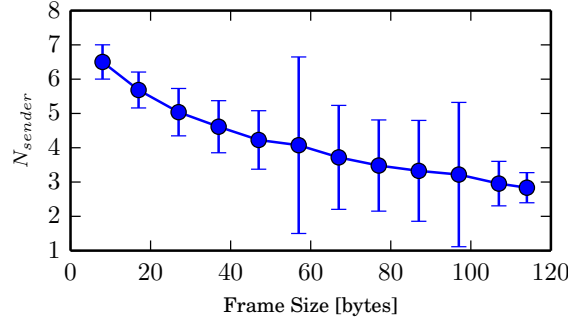


FIGURE 4.3 – Nombre moyen de tentatives d’envoi en fonction de la taille de la trame.

Verifier que les notations sont consistantes avec celles du chapitre sur le cache

$$S_{cost}(m) = P_{TX}N_{sender}(m)T_p(m) + P_{RX}t(ACK) \quad (4.1)$$

$$R_{cost}(m) = P_{RX}N_{receiver}(m)T_p(m) + P_{TX}t(ACK) \quad (4.2)$$

Dans le reste du chapitre nous utiliserons $S_{cost}(m)$ pour désigner l’énergie pour envoyer une trame de $s(m)$ octets. Cette énergie prends en compte la procédure d’accès au canal et le préambule de transmission si il y en a un. De même, $R_{cost}(m)$ désigne l’énergie dépensée par un nœud pour recevoir une trame de $s(m)$ octets.

où $N_{sender}(m)$ et $N_{receiver}(m)$ sont le nombre de tentatives entreprises par l’expéditeur et le destinataire respectivement.

Dans le reste du scénario, nous considérons que le trafic réseau est suffisamment faible pour que les collisions soient négligeables. Cependant, en raison de leurs large périodes d’endormissement, une désynchronisation des nœuds est possible et un émetteur et un récepteur ne seront pas forcément réveillé au même instant ce qui peut engendrer du strobbing.

4.2.2 Strobbing en CSMA/CA

En effet, avec ContikiMAC, un expéditeur doit transmettre plusieurs fois une trame avant de recevoir un acquittement. Même avec des mécanismes de verrouillage de phase de ContikiMAC, les horloges dérivent et plusieurs tentatives peuvent être nécessaire pour l’envoi d’une trame. Nous évaluons cet effet en simulation en mesurant combien d’essais sont nécessaire en moyenne avec seulement deux nœuds. Dans cette expérience puisqu’il n’y a qu’un expéditeur et un seul destinataire, on évite les collisions de paquets d’un tiers. Nous comptons combien de fois un paquet est envoyé par l’expéditeur avant d’être acquitté par le destinataire.

La figure 4.3 représente la moyenne du nombre de transmission $N_{sender}(m)$ nécessaire à l’envoi d’une trame de $s(m)$ octets entre deux nœuds. Les mécanismes de CSMA/CA et les cycles de sommeil ajoutent des couts à la transmission théorique d’une trame de taille donnée. Pour mesurer ses couts, nous effectuons une simulation avec une source et une destination qui échangent du trafic. Comme nous l’avons vu dans la section 2.4.1, ContikiMAC induit souvent le fait d’envoyer plusieurs fois un paquets avant qu’il ne soit acquitté. Ces multiples tentatives ne peuvent pas être prédite facilement pour un nœud à un instant précis cependant des comportement moyens peuvent être connus pour une taille de paquet donnée.

Dans la figure 4.3, le nombre d’envoi nécessaire pour envoyer un long paquet est plus faible que pour en envoyer un court. C’était à prévoir sachant que les trames longues mettent plus de temps pour

être transmises et ont donc plus de chance d'être écouté pendant un réveil du destinataire. En outre, nous pouvons dire que c'est l'expéditeur qui va dépenser le plus d'énergie dans cette configuration. Comme les paquets ACK ont une taille constante, le cout de réception pour le destinataire est constant.

De son côté, le destinataire se réveille en moyenne au milieu d'une transmission et attends le prochain essai d'envoi pour recevoir la trame complètement et envoyé son acquittement. Le nombre de trames reçu peut être estimé à $N_{\text{receiver}} = 1.5$. La passerelle n'ayant pas de cycles de veille, pour les nœuds en contact direct avec elle on a $N_{\text{sender}}(m) = 1$. Nous remarquons également une croissance linéaire, ce qui est attendu puisqu'une plus le paquet est loin plus il est nécessaire de rester en réception pour le recevoir intégralement. La raison pour laquelle le cout pour la destination est moins élevé que pour la source vient du fait que la source a besoin de nombreux essais pour transmettre avec succès alors que dans le cas de la réception il faut moins de deux tentatives.

Ce cycle de veille est basé sur IEEE 802.15.4 comme nous l'avons vu dans 2.1 et utilise CSMA/CA pour éviter les collisions et écoute toujours quand la radio ne transmet pas afin de garantir une certaine stabilité du réseau.

Un nœud peut être le destinataire d'un paquet 6LoWPAN mais peut aussi être utilisé comme relais dans un chemin multi-sauts. La topologie devient donc un paramètre déterminant pour savoir si un nœud passera beaucoup de temps à relayer des paquets pour ses voisins et donc d'utiliser une grande partie de ses ressources à cette fin.

- On envoie la liste des voisins en utilisant une combine avec des filtres de bloom. Peu importe si les filtres de bloom sont vraiment implémentés dans le réseau. Ce qui compte c'est de voir si la passerelle serait capable de les retrouver si elle venait à recevoir un message de ce type. On gère les voisins en utilisant des filtres de bloom pour envoyer la liste des voisins. Si un nœud fait partie des voisins du nœud A alors on calcule le hash de son adresse IPv6 puis on l'ajoute dans le filtre de bloom. Une fois que la passerelle récupère l'information elle est en mesure de tester vis à vis de tous les nœuds qu'elle connaît de savoir si ils interagissent ou pas avec une certaine probabilité.
Comment s'assurer que l'on a un envoi de message efficace ?
- Comment ça se passe quand on est en TDMA façon TFSCH par rapport à une utilisation façon CSMA quand on est dans Contiki MAC ?

4.3 Supervision passive

Nous considérons une expérience de supervision typiques dans laquelle les nœuds utilisent ContikiMAC [32], qui est un mécanisme de cycle de veille et RPL [114] qui est un protocole de routage. Grâce à des simulations dans COOJA [81] nous évaluons les différentes stratégies et comparons les résultats d'estimations avec ceux disponibles par le simulateur. Les résultats montrent que la connaissance des routes améliore significativement les estimations.

En combinant cette approche à une approche de supervision active nous démontrons comment la passerelle peut apprendre les biais induits de la supervision passive et qui sont dus à des phénomènes non inféribles à la passerelle comme des pertes de paquets dus à des conditions radio. Nous pouvons ainsi corriger les estimations de supervision passive en conséquence.

Notre but est d'étudier les possibilités de la passerelle à superviser de manière passive le trafic réseau. la consommation énergétique des nœuds en regardant le trafic réseau qu'elle gère. Comme les communications radios sont les principales sources de consommation énergétique.

4.3.1 Supervision passive du trafic réseau

En utilisant une estimation du cout d'échange de trames entre deux nœuds, nous pouvons prédire au niveau de la passerelle, l'impact énergétique du trafic réseau pour tous les nœuds. Nous introduisons deux estimateurs utilisant des données différentes pour apprendre.

4.3.1.1 Supervision passive sans topologie

L'estimateur naïf le plus simple n'estime l'impact que sur la source et la destination des paquets. Dans le scénario présenté sur la figure. 4.2 dans lequel un nœud E envoie une trame à la passerelle G , l'estimateur naïf ne déduit que l'énergie consommée par les E et G . Tous les autres nœuds appartenant au réseau sont ignorés.

Soit \mathcal{D}_i les messages provenant de i et \mathcal{A}_i les messages autant pour destination finale i . Nous obtenons l'énergie estimée suivante :

$$\begin{aligned} S_i(t) &= \sum_{m \in \mathcal{D}_i(t)} S_{\text{cost}}(m) \\ R_i(t) &= \sum_{m \in \mathcal{A}_i(t)} R_{\text{cost}}(m) \\ \hat{E}_i(t) &= \sum_{m \in \mathcal{D}_i(t)} S_{\text{cost}}(m) + \sum_{m \in \mathcal{A}_i(t)} R_{\text{cost}}(m). \end{aligned}$$

Cet estimateur est adapté pour des topologies en étoile à un saut et peut être déployé sans engendrer de paquets de supervision supplémentaires. Cependant il ignore les couts engendrés dans des scénarios multi-sauts par les retransmissions de paquets.

4.3.1.2 Supervision passive avec topologie

Dans des topologies multi-sauts, la passerelle connaît le routage au sein du LLN. Les protocoles de routage comme RPL connaissent les routes descendantes depuis la passerelle et peuvent utiliser ces connaissances pour construire une représentation complète de la topologie réseau. Ainsi nous présentons le second estimateur présenté comme *Route* et utilisant les informations de topologies pour déduire l'énergie consommée par le transfert des messages comme montré en gris sombre sur la figure 4.2).

Soit \mathcal{F}_i l'ensemble des trames qui sont retransmises par le nœud i alors qu'il n'est ni la source ni le destinataire du message.

$$\hat{E}_i(t) = \sum_{m \in \mathcal{D}_i(t) \cup \mathcal{F}_i(t)} S_{\text{cost}}(m) + \sum_{m \in \mathcal{A}_i(t) \cup \mathcal{F}_i(t)} R_{\text{cost}}(m)$$

4.3.2 Résultats expérimentaux - Topologie en chaine

Nous avons évalué la précision des estimateurs en utilisant le simulateur COOJA avec son extension Powertracker qui mesure le temps que chaque nœud émulé passe dans un état de consommation énergétique précis avec une résolution de $1\mu s$. Les nœuds utilisaient Contiki comme système d'exploitation, RPL comme protocole de routage et ContikiMAC sur IEEE 802.15.4 .

Considérons une topologie simple à 7 nœuds comme représenté sur la figure 4.4. Les nœuds ne peuvent envoyer et recevoir de paquets que de la part de leurs voisins adjacents. Chaque nœud envoie à destination de la racine un message de 10 octets par seconde ce qui induit une trame de 69 octets

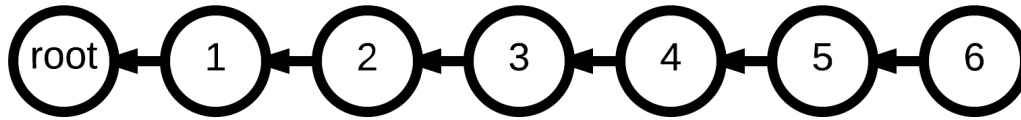


FIGURE 4.4 – Topologie réseau

Tension d'alimentation	3.6 V
MCU on, Radio RX	21.8 mA
MCU on, Radio TX	19.5 mA
MCU on, Radio off	1800 μ A
MCU standby	5.1 μ A
MCU idle, Radio off	54.5 μ A

TABLE 4.1 – Consommation énergétique du Tmote sky

sur le canal. Dans ce scénario durant 200 secondes, chaque nœud envoie à la racine un paquet UDP avec une charge utile de 10 octets chaque seconde. Ainsi on obtient pour résultat une trame de 69 octets sur le canal. Nous utilisons une valeur moyenne de strobbing de 3.76 comme trouvé dans les expériences précédentes de calibration dans la section ???. Nous utiliserons les données présentées dans la Table 4.1 pour calculer l'énergie consommée.

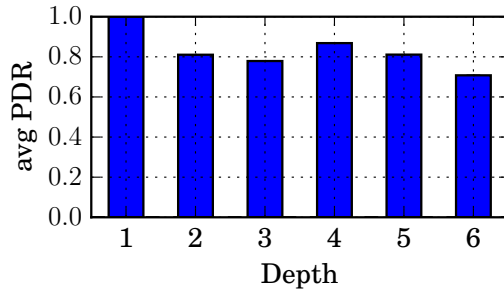
Les nœuds les plus proches d'une racine doivent relayer plus de trafic en provenance des nœuds sous-jacents en plus de leurs propres trafic.

4.3.2.1 Analyse de l'impact de la profondeur

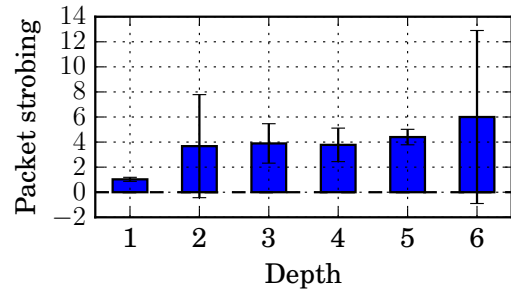
La figure Fig. 4.5a représente le ratio de succès de transmission de paquets pour chaque nœuds en fonction de la distance à la racine. Nous pouvons voir que le ratio de paquets acheminés n'est pas uniforme sur tout le réseau et que les nœuds qui ne sont pas connectés directement à une racine souffrent de congestions et de collisions. La figure 4.5a nous révèle la proportion du trafic que la passerelle peut effectivement voir.

Il y a plusieurs difficulté pour estimer le trafic réseau. La première c'est qu'il y a du trafic que la passerelle ne peut pas voir passivement. De plus les pertes de paquets sont fréquentes notamment loin de la passerelle. Ces pertes de paquets sont causées par des congestions fréquentes dans une topologie comme celle de la chaîne. Les nœuds proches de la racine doivent gérer les paquets de tous les nœuds éloignés et ils ont de bonnes chances d'être perdus.

La figure 4.5b représente le nombre moyen d'envois de paquets i.e. le nombre de tentatives que vont faire chaque nœud pour envoyer une trame à son parent. Cette métrique quantifie le cout d'envoi d'une seule trame sur chaque saut. Cette métrique quantifie le cout d'envoi d'une trame sur chaque saut. Nous remarquons une corrélation nette entre le nombre de sauts à la distance à la racine et cette courbe. Nous pouvons voir que le nombre d'envoi moyen augmente alors que l'on s'éloigne de la racine.



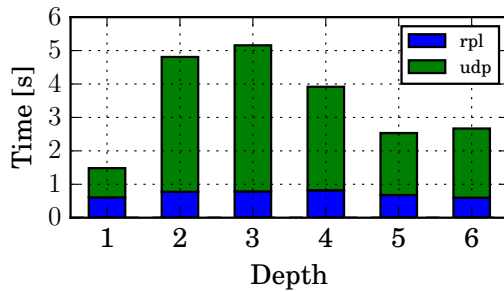
(a) Packet Delivery Ratio (PDR).



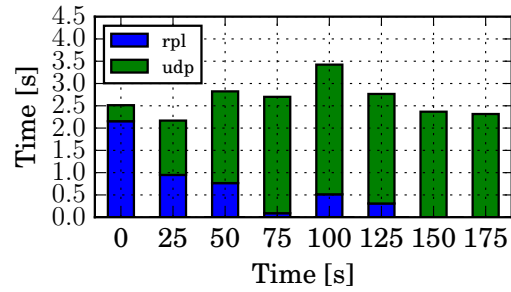
(b) Tentatives d'envois de paquets par profondeur.

4.3.2.2 Répartition et évolution des protocoles

La figure. 4.6a représente la distribution des types de trafic se présentant sur chaque saut (Message de routage et UDP). Ces mesures montrent l'importance du contrôle de trafic que la passerelle ne mesure pas et devraient inférer. Nous remarquons qu'à ce stade, le trafic est presque identique pour tous les nœuds et pourrait être modélisé par un flux constant. Cependant, la figure 4.6b montre l'évolution de ce phénomène. Nous pouvons observer un gros volume de paquets de routage requis pour construire le DODAG de RPL. Une fois cette phase terminée, le mécanisme de Trickle permet de réduire la quantité de paquet de routage émis car le réseau est stabilisé. La majorité du trafic est maintenant du au trafic applicatif. Cela confirme que la construction et la réparation des structures de routage génèrent un large trafic difficile à prévoir. Cependant la pertinence de l'estimation basée sur le trafic applicatif est bien motivée lorsque le réseau est dans un état stable.



(a) Répartition des protocoles par profondeur.



(b) Évolution de la répartition des protocoles.

TODO : Voir les résultats

Comparons à présent les estimations aux valeurs réelles. La figure ?? représente le rapport entre le temps estimé et le temps réellement passé en transmission et réception pour les nœuds 3 et 4.

Le nœud 7 ne relaie aucun paquet, car il se trouve au bout de la chaîne. Les nœuds 3 et 5 relaient une part importante de trafic des nœuds sous-jacents. Le nœud 2 a un rôle similaire au nœud 3, cependant comme la racine est toujours en écoute sur le canal, le nœud 2 perd moins d'énergie lors de tentatives infructueuses. La position des nœuds importe donc et ces résultats le démontrent.

Nous pouvons voir que l'estimateur *naïf* sous-estime largement l'activité des nœuds. Ce phénomène est attendu car l'estimateur *naïf* ignore le coût de retransmissions des nœuds intermédiaires. Lorsque la connaissance des routes est utilisée, l'estimation est meilleure mais toujours en dessous.

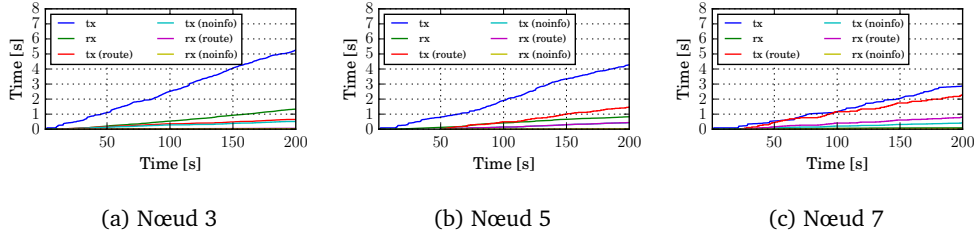


FIGURE 4.7 – Cout estimé dans les scénarios de chaines

des valeurs réelles. La sous évaluation initiale est expliquée par la phase de construction du DODAG comme illustré sur la figure 4.6a. Le trafic “applicatif” ne démarre sur chaque nœud que quand RPL a trouvé une vers la passerelle. Cependant les différences restent non négligeables même après que la topologie de routage ait été établis ce qui signifie que l’estimation passive n’est pas suffisante, un mécanisme d’appoint reposant sur des messages explicite est nécessaire pour avoir une supervision efficace.

Il faut aussi dire que si le biais est constant, on peut multiplier par un facteur d’échelle. Si les tendances sont linéaires dans les deux cas notre estimation est correcte a un facteur multiplicatif près.

Cette simulation confirme que l’évaluation du trafic n’est pas complètement faisable sans possibilité de corriger les estimations par des mesures précises.

4.3.3 Supervision active & passive

Comme nous l’avons vu dans les parties précédentes, la supervision passive permet de prévoir une partie de la consommation énergétique des nœuds. Lorsqu’elle est disponible, la supervision active permet d’avoir des relevés plus précis. Notre but ici est de permettre de réduire le nombre de messages de supervision active nécessaire pour avoir une bonne vue du réseau.

Nous évaluons le processus de supervision active en considérant une topologie réseau représentée sur la figure 4.8. Cette topologie est composée de 21 nœuds clients envoyant des paquets à la racine chaque seconde. Nous avons pris la valeur de $\alpha = 0.25$ afin de ne pas réagir trop brusquement aux pics de trafic brusques causés par une reconstruction du DODAG RPL.

TODO : La vision periodique c’est la méthode la plus naïve et celle contre laquelle on se bat justement. Est ce que ça vaut pas le coup d’essayer de faire d’autres méthodes ? On a essayé avec des bandits manchots est ce que l’on peut avoir d’autres possibilités ?

Nous considérons des messages de supervisions explicites envoyés par le nœud vers la passerelle. Les messages contiennent les temps mesurés par le nœud passé dans chaque état de transmission. Notre objectif est de trouver quel est le rythme optimal d’envoi en fonction des conditions réseau. Quand la passerelle reçoit un message de supervision explicite au temps t elle recalcule son estimateur de consommation d’énergie pour le nœud $\hat{E}(t)$ de la manière suivante :

$$\hat{E}(t) = E(t_r) + R_i(t) + S_i(t) + \epsilon(t_r) \frac{(t - t_r)}{T} \quad (4.3)$$

$$\epsilon(t_r) = \alpha.(E(t_r) - \hat{E}(t_r)) + (1 - \alpha).\epsilon(t_{r-1}) \quad (4.4)$$

où t_r et t_{r-1} sont les deux dernières dates de message de supervision active. S_i et R_i sont respectivement les couts estimés d’envoi et de réception induis par le trafic applicatif depuis le dernière

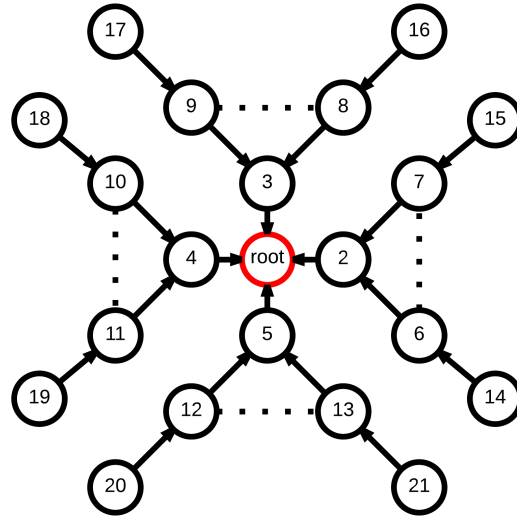


FIGURE 4.8 – Topologie réseau et radio.

supervision active et $E(t_r)$ est le niveau d'énergie consommé réel du nœud à t_r . $\epsilon(t_r)$ est l'estimation de l'erreur apprise des précédents messages de supervision active. Il doit intégrer les consommation énergétique manquée par notre estimateur de base comme le trafic non routés vers la passerelle ou bien les pertes de paquets. Il peut être calculé en utilisant une Exponentially Weighted Moving Average (EWMA), comme montré dans l'équation TODO montrer cette équation. Au temps t , cette erreur est prise proportionnellement au temps depuis la dernière supervision active qui se produit tous les T .

TODO : Ajouter une mention à l'équation explicitement.

$\frac{E(t_r) - E(t_{r-1})}{t_r - t_{r-1}}$ représente la tendance de l'énergie consommée et prends en compte les paquet émis par le réseau y compris ceux que la passerelle n'a pas pu prévoir. Notons que $\epsilon(0) = 0$.

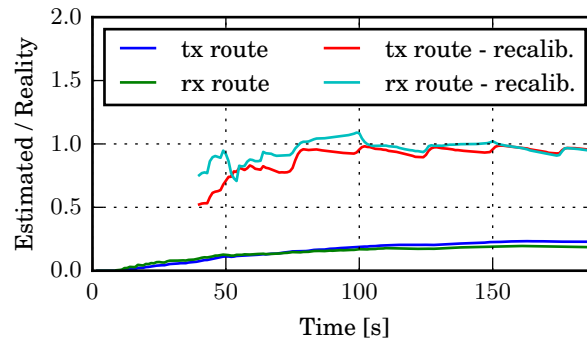


FIGURE 4.9 – Erreur relative pour la topologie réseau avec une supervision active ($T = 25s$).

Comme nous pouvons le voir sur la figure 4.9, la supervision active périodique fournit la mesure correcte. Malgré la divergence observée au début du à la construction des structures de routage comme expliqué dans la section 4.3.2.

Les messages de supervision active qui améliore la précision de l'estimation, mais qui a un cout

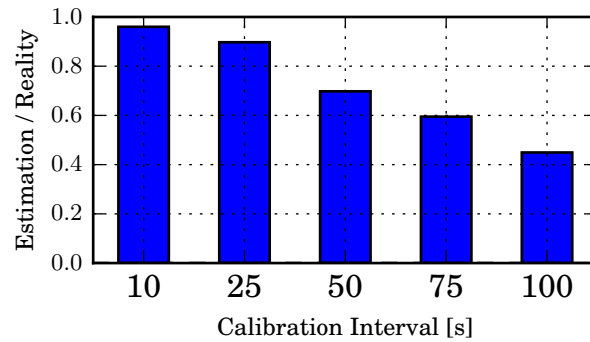


FIGURE 4.10 – Erreur relative pour différents intervalles de supervision active.

linéaire avec les intervalles de de supervision active et devrait être aussi réduit que possible. Fig. 4.10 représente le ratio moyen obtenu par l'estimateur utilisant la topologie de routage *Route* sur une simulation de 200 seconds en fonction recalibration events interval.

Afin de réduire le cout des messages de supervision devrait être réduit au cours du temps à mesure que le réseau se stabilise.

Dire que notre processus de mise a jour est cyclique on va voir les nœuds a chaque instant et on a pas de processus de lissage.

Nous observons que les estimations de transmission divergent beaucoup plus que les estimations de réception. C'est cohérent avec les résultats montrés dans la figure 4.3. La couche MAC est asynchrone, ainsi les nœuds passent plus de temps à transmettre qu'à recevoir. Ainsi les biais d'estimation ont plus d'impact pour la transmission que pour la réception.

Combien de temps me faut il pour me rendre compte qu'un nœud est indisponible ? Cela dépends de leur cycle de sommeil, cependant si une requête applicative n'est pas satisfaite au bout d'un temps donné, la passerelle peut considérer que le nœud est indisponible.

Combien de temps me faut il pour détecter une panne ? Cela dépends de la politique de supervision, dans les cas à événements le temps dépends de beaucoup d'autres facteurs. Dans un cas cyclique il correspond dans le pire cas au temps T . C'est au bout de ce temps que l'on peut savoir si notre modèle est confirmé par l'expérience ou non.

Quel est le délai pour détecter un gros problème dans le réseau ? La corrélation entre les différentes métriques de supervision est laissée en dehors des problématiques de nos contributions.

Comment trouver les fréquences d'envoi de messages de monitoring optimaux ? C'est strictement lié à la politique de supervision et le temps avec lequel on peut tolérer qu'un nœud ne donne pas de signes de vie. Dans le cas où la bande passante, les ressources et les fonctionnalités sont disponibles et suffisantes, on peut adopter des méthodes classiques.

4.4 Conclusion

Dans ce chapitre, nous avons introduit un mécanisme d'estimation fonctionnant au niveau de la passerelle, utilisant le trafic qui y passe pour évaluer la consommation énergétique individuelle des nœuds.

Nous montrons, à travers des simulations effectuées avec COOJA, que l'utilisation d'informations déjà disponibles à la passerelle permet d'améliorer la performance des estimateurs de trafic. Cependant, seulement en regardant le trafic passant par la passerelle ne suffit pas pour avoir une estimation précise. Un mécanisme de mise à l'échelle est nécessaire de tenir compte des cycles d'endormissement

et de couche MAC, puisque l'intervalle de sommeil introduit une désynchronisation entre émetteurs et récepteurs, qui se traduit par multiple transmission tente pour chaque trame.

Même dans le cas d'un trafic à débit constant, les nœuds doivent envoyer explicite rapports de l'énergie, que nous appelions supervision active, à la passerelle, au moins dans la phase d'initialisation du réseau. Ceci est nécessaire pour prendre en compte l'impact du routage des paquets échangés pour construire la topologie réseau initiale. Dans le cas d'un réseau statique et d'un trafic régulier, ces messages de supervision explicites deviennent rapidement pas nécessaire. En présence d'un modèle de trafic irrégulier, nous pensons que la dynamique peut être capturé par le passerelle, comme il est généralement la source ou la destination du trafic. Les travaux à venir consisteront à étudier l'effet de la dynamique du réseau en tester les modèles de trafic irréguliers.

Voir a changer le nombre de messages explicites en fonction de multiples critères.

Qu'est ce qui se passe avec un trafic non constant et non régulier ?

Parler des bandits manchots.

Quelles sont les topologies significativement difficiles à estimer ? Est ce qu'il y a une corrélation entre topologie et estimations ? Est ce que ça fonctionne avec des topologies aléatoires ? Changeante (Ajout ou perte de nœuds) ?

Chapitre 5

Expériences automatisées et reproductibles

It doesn't matter how beautiful your theory is, it doesn't matter how smart you are. If it doesn't agree with experiment, it's wrong.

Richard Feynman

Contents

5.1 Makesense	56
5.1.1 Justifications	56
5.1.2 État de l'art & Écosystèmes	57
5.1.3 Architecture	57
5.2 Construction d'une expérience reproductible	59
5.2.1 Fabrication	59
5.2.2 Déploiement - Execution - Déplacement des traces	59
5.2.3 Mise en forme des résultats intermédiaires	60
5.2.4 Analyse des résultats	61
5.2.5 Présentation des résultats	61
5.2.6 Garanties de reproductibilité par intégration continue	62
5.3 Conclusion & Perspectives	63

Peut être qu'il faut plus parler de répétabilité et dire la différence avec la reproductibilité.

Contribution Démonstration d'une méthodologie reproductible et automatisée d'expérience sur réseau de capteurs démontrant la possibilité d'utilisation de Contiki et Cooja pour avoir une méthode de développement local utilisant l'émulation puis de déployer le même code sur Iotlab et d'avoir des cycles d'itérations courts entre l'implémentation et le test sur système réel.

Nous présenterons dans ce chapitre la méthodologie que nous avons utilisée pour obtenir les résultats précédents et nous introduirons Makesense [67]. Il faut annoncer le plan

5.1 Makesense

La conception et la validation de protocoles efficaces peuvent être réalisées par simulateurs et pas bancs d'essai réalistes. Cependant, les deux approches présentent des inconvénients. Dans le cas des simulateurs, il est possible d'effectuer l'évaluation de la performance pour un très grand nombre de nœuds dans un laps de temps raisonnable. Cependant, les simulateurs comme font plusieurs hypothèses sur le système physique, la nature des transmissions et plus largement sur la fiabilité du système. Les bancs d'essai réels sont quant à eux garants d'une analyse de performances très précise. Cependant, le passage à l'échelle d'expériences nécessite souvent un effort pour le déploiement du micrologiciel, la collecte des résultats et l'analyse de ces derniers qui s'accompagnent souvent de problèmes subtils et chronophages.

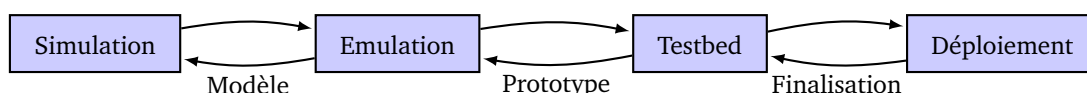


FIGURE 5.1 – Cycles de développement

Les cycles de développement usuels représentés sur la figure 5.1, montre qu'il est assez courant de partir d'un modèle pour aller vers des implémentations réelles. Cependant, dans de nombreux cas, le cout de passage d'une étape à l'autre est élevé. Le but de ce chapitre est de montrer comment chaque phase de développement et d'analyse peut être automatisée afin de factoriser un maximum d'étape de développement.

5.1.1 Justifications

Contraintes des expériences sur les réseaux de capteurs Cooja [81] est le simulateur que nous utilisons durant nos phases de développement. Il permet de rapidement tester si un micrologiciel fonctionne et si dans des hypothèses raisonnables, nos réseaux se forment. Il est aussi possible de visualiser précisément les cycles de veille, la consommation énergétique estimée et de pouvoir inspecter un nœud durant une simulation. Le contrôle sur l'environnement de simulation est total et détaillé. Si les graines des générateurs de nombre aléatoires sont fixées alors on a une reproductibilité complète de l'expérience. Cependant, Cooja fait un certain nombre d'hypothèses sur le médium de transmission et plus généralement sur le fonctionnement des nœuds, la dérive des horloges, l'usure des composants et les pannes éventuelles. De ce fait, le réalisme de l'expérience est toujours modéré.

À l'inverse dans le cas d'un banc de test, le réalisme est fort, les problèmes d'intégration sont nombreux, variés et difficilement prévisibles. Le temps ne peut pas être accéléré et des expériences longues sont nécessaires pour tester la fiabilité des nœuds. Certaines plateformes comme Iotlab offrent des traces détaillées de l'expérience, mais aucune ne peut garantir que deux expériences donneront les mêmes résultats. Enfin même s'il est possible de contrôler les nœuds assez finement dans une certaine mesure, il n'est pas toujours possible de contrôler l'environnement radio et l'atténuation des signaux.

Comme montré dans la figure 5.2, en phase de simulation, il est attendu d'avoir des preuves de concepts rapides et un cout de cycle d'itération faible. En phase d'expérimentation réelle, il est attendu d'avoir des preuves de robustesse des systèmes et les cycles d'itération sont beaucoup plus longs.

Automatisation Les expériences et les simulations peuvent être décomposables en plusieurs sous parties. Les mêmes transformations sont effectuées d'une expérience à une autre. Ne pas les automatiser les rend plus susceptibles aux erreurs de configurations et aux oublis.

Recherche reproductible La reproductibilité est un des piliers de la méthode scientifique [89, 39]. Cependant, ce volet est rarement mis en avant, aussi bien au moment de juger un article pour une

	Realisme	Contrôle	Itération rapide
Testbed (Iotlab)	✓	✗	✗
Simulation (Cooja)	✗	✓	✓

FIGURE 5.2 – Évaluation des solutions expérimentales

revue ou bien lorsqu’il est publié. Ce paradoxe a été relevé à de nombreuses reprises dans d’autres sciences [85, 113] et il est d’autant plus surprenant en informatique où les coûts de reproduction d’une expérience sont souvent modestes. Le développement des bancs de test et des outils de simulations rendent la reproductibilité plus abordable. Makesense a pour objectif de contribuer à améliorer la situation en proposant une série d’outils issue du monde logiciel comme l’intégration continue pour démontrer la reproductibilité d’une expérience.

5.1.2 État de l’art & Écosystèmes

Survey sur l’automatisation de testbed [13].

Automatiser les tâches récurrentes est un processus courant en informatique. Cependant, les bibliothèques logicielles qui ont pour tâches spécifiques de formaliser intégralement une expérience sur les réseaux de capteurs sont manquantes. Il n’existe pas d’outils complètement intégrés permettant de gérer le cycle complet d’une expérience orientée capteur et de la déployer sur un banc de test. Nepi [61] propose l’idée d’abstraire le banc de test au profit d’objets génériques. Le problème de cette approche est qu’elle repose essentiellement sur des adaptateurs qui doivent être disponibles pour chaque banc de test. Cooja [81] et Iotlab [40] étaient absents de la liste des adaptateurs proposés.

Plutôt que de créer un outil définitif ayant pour tâche étroite de gérer toutes nos simulations. Nous avons privilégié la réutilisation d’outils compatibles entre eux et l’orchestration de ces derniers. Utiliser un écosystème de logiciels ayant des communautés larges et ancrées dans les problématiques scientifiques est un choix que nous jugeons plus pertinent à celui de construire un Domain Specific Language (DSL) pour résoudre un seul type de problème.

Indiquer également que les contraintes de l’expérimentation dans les réseaux de capteurs et plus généralement dans l’embarqué sont plus simple que dans le monde des testbeds partagés. Typiquement, le contrôle sur un noeud est total dès lors que la réservation est faite. Ainsi les problématiques de virtualisation et de mutualisation d’équipements n’existent pas.

5.1.3 Architecture

Makesense rassemble les paramètres, fonctions et résultats dans un seul et même Jupyter notebook. Il s’agit d’un fichier texte au format JSON. Il contient l’intégralité des variables, fonctions et algorithmes que nous allons utiliser au sein de l’expérience.

5.1.3.1 Présentation de Jupyter et de son notebook

Jupyter (anciennement IPython [86]) est une plateforme libre pour l’informatique interactive et parallèle. Jupyter utilise le concept de notebook pouvant être vu sur un navigateur web classique. Ce notebook combine code, texte, des expressions mathématiques et des fonctions interactives ainsi que d’autres médias riches au sein d’un document web partageable.

Lorsqu’une cellule de texte riche est rendue, le texte et les formules mathématiques au format \LaTeX contenu dans cette cellule sont rendus à l’utilisateur. Ainsi il est possible d’introduire des formules mathématiques complexes et d’obtenir un rendu fonctionnant sur n’importe quel navigateur moderne.

Lorsqu'une cellule de code est rendue, elle est exécutée dans le noyau. Ce noyau est par défaut en Python. Le contexte de chaque cellule est passé au noyau et de cette manière. Ainsi, une fois exécutées, les variables sont gardées dans le contexte du noyau et sont partagées à travers toutes les cellules du notebook.

Du point de vue de Makesense, une expérience est découpée en plusieurs sous parties qui peuvent être ré exécutées. Pour ce faire, le code est découpé et stocké au sein des cellules du notebook.

Comme ce fichier texte rassemble l'ensemble des paramètres, il est facile de le modifier et de l'utiliser comme modèle pour gérer d'autres expériences au sein d'une même campagne de simulations. Puisque chaque notebook n'est qu'un fichier texte avec un format connu, il est possible de générer à la chaîne des expériences complètes en s'inspirant des méthodes que nous allons voir dans la Section 5.2.1.

5.1.3.2 Interactions entre les outils

Utiliser un seul et même langage pour organiser nos expériences et traitements est un atout. Plutôt que de chercher à coller des composants hétéroclites utilisant différents langages, nous avons privilégié les solutions ouvertes, légères et agnostiques. Python dispose de cette versatilité et possède une communauté ancrée dans la recherche scientifique. En outre lorsqu'une bibliothèque est manquante ou insuffisante il est toujours possible d'interfacer un outil externe et de l'automatiser. Enfin, Jupyter découple le moteur d'exécution du format du JSON du notebook. Ainsi il est possible d'utiliser un moteur différent tel que Julia ou bien R.

5.1.3.3 Découpage en étapes et cellules

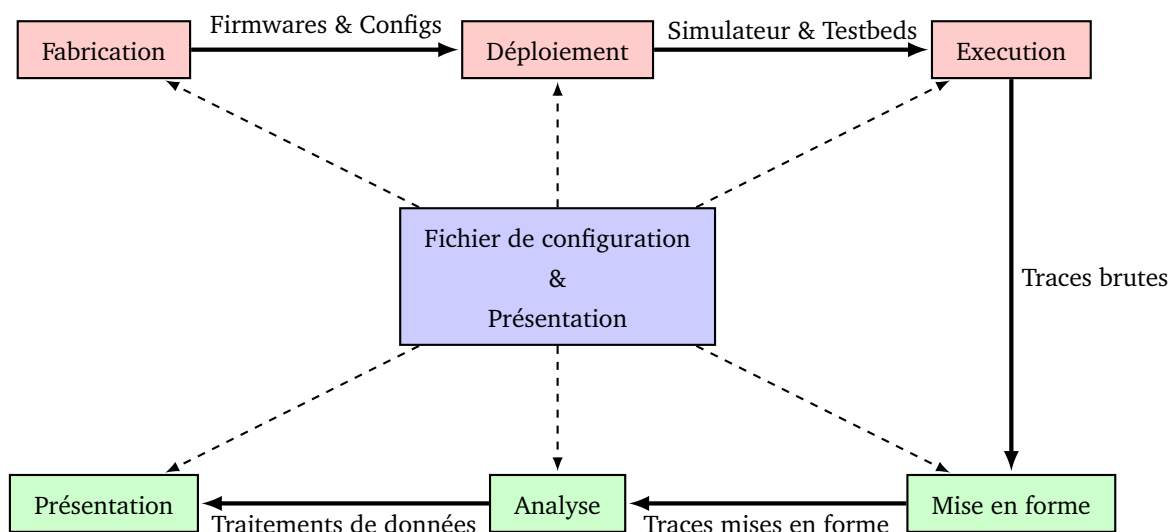


FIGURE 5.3 – Schéma général des étapes dans Makesense

Du point de vue de Makesense. Une étape d'une expérience peut être découpée en cellule de notebook. Ainsi, il est possible de ré exécuter certaines cellules indépendamment des autres. En outre, documenter chaque étape avec du texte riche et des images rend le processus expérimental beaucoup plus clair et interactif.

5.2 Construction d'une expérience reproductible

Afin d'illustrer l'intérêt de Makesense et plus généralement de l'écosystème Python, nous allons exposer comment organiser une expérience utilisant Contiki [30] et le simulateur Cooja [81].

5.2.1 Fabrication

Description de l'étape Cette étape de préparation compile les systèmes pour les nœuds contraints et produit tous les fichiers nécessaires à la simulation. Cette étape est non triviale, en effet, il est courant d'être confronté à différentes plateformes matérielles tout au long de l'expérience. Dans le cas de Contiki, il est courant de développer sur des plateformes de types MSP, car Cooja permet de facilement les émuler. En revanche dans le cas d'Iotlab il est courant d'utiliser une plateforme différente comme les cortex M3.

Dans le cas de simulation, Cooja requiert la création d'un fichier principal qui va être utilisé pour placer les nœuds, les démarrer et interagir avec eux par un script JavaScript. Cette étape se charge de générer des positions de nœuds automatiquement et d'affecter les micrologiciels aux nœuds correspondants. En outre, les fichiers de configurations obtenus utiliseront la graine de générateurs aléatoires et les portées de transmission et réception voulues et configurées.

Méthodes & outils utilisés Jinja2 [87] est un moteur de modèle type qui permet de générer des fichiers textes à partir de modèles. Ce moteur est particulièrement utile pour générer une série de fichiers sources avec différentes variables/programmes pour le système de nos nœuds. Des extraits de code se trouvent dans l'annexe A.1.

Justification de cette méthode Un cas d'usage classique de cette étape est de créer une *campagne* de simulations avec des nœuds ayant des fonctionnalités et différents paramètres codés en dur. Dans le cas de nœuds très contraints, il est possible de générer un code source arbitraire et de diminuer la gestion dynamique des arguments.

Comparaisons avec les autres méthodes / Etat de l'art Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

5.2.2 Déploiement - Execution - Déplacement des traces

Description de l'étape Lors du déploiement d'un nœud sur un banc de test particulier, il est courant de devoir faire correspondre une table de correspondance entre les nœuds et les endroits où ils vont être installés. De même, il peut arriver que l'on désire régénérer à la volée cette table de correspondance en cas d'indisponibilité d'un banc de test. Le fait de générer à la volée cette table facilite cela. De plus, il est courant de devoir gérer plusieurs tâches en concurrence telles que la génération de trafic, le lancement des processus d'agrégation de journaux ainsi que la vérification et la supervision des connexions. Il s'agit de gérer les pannes et erreurs qui peuvent advenir tout au long de la simulation. Ces tâches sont asynchrones et dans le cas d'un banc de test le fait de lancer une expérience ne veut pas nécessairement dire que l'expérience va être effectuée immédiatement. Ainsi il est utile d'automatiser tous ces processus afin de ne rien oublier à chaque lancement.

Méthodes & outils utilisés Fabric [41] est une bibliothèque permettant de transmettre et d'exécuter des programmes sur des plateformes distantes. Depuis la ligne de commande, il va être possible de lancer différentes fonctions codées en Python. Dans le cas du banc de test Iotlab [40], l'envoi et la récupération des différentes traces ont été automatisés afin de pouvoir reproduire n'importe quelle expérience lancée. Des extraits de code sont disponibles en Annexe A.2.

Justification de cette méthode Il serait possible de lancer des commandes de manière séquentielle et obtenir le même résultat. Cependant, intégrer directement au sein de nos scripts des moyens de régénérer à la volée des fichiers de configurations différents selon le banc de test utilisé est une grande aide en cas d'indisponibilité d'une plateforme. En outre, la possibilité de récupérer parallèlement les résultats obtenus sur différents bancs de test accélère considérablement les tâches de déplacements des traces vers les endroits où ils seront traités.

Enfin, abstraire la localisation de nos traces permet d'effectuer des traitements systématiques, peu importe l'endroit où les résultats ont été produits. Ainsi, on peut passer plus rapidement à la prochaine étape de mise en forme.

Comparaisons avec les autres méthodes / Etat de l'art Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

5.2.3 Mise en forme des résultats intermédiaires

Description de l'étape Cette étape est lancée une fois que les traces de l'expérience ont été récupérées. Il s'agit de fichiers Packet CAPture (PCAP) et de multiples journaux textes représentant les événements qui se sont produits.

Certaines transformations que nous effectuons sur les traces sont triviales comme des conversions d'unités et des normalisations de valeurs. D'autres peuvent être la transformation des adresses MAC vers des identifiants plus facilement compréhensibles et pouvant nous aider à faire une cartographie des événements dans le réseau. Enfin, dans le cas de système très contraint, il est possible que des textes intelligibles soient remplacés par des écritures condensées à quelques caractères qui rendent la lecture des résultats plus difficile. Cette étape permet de retransformer toutes les traces "compressées" vers des noms plus détaillés.

Nous classifions certains événements apparaissant des traces. Dans le cas d'un trafic réseau, il est nécessaire de pouvoir classer chaque paquet selon une série de filtres qui vont nous permettre dans la prochaine phase d'analyse d'effectuer des traitements quantifiés. Effectuer des assainissements en fonction du type de paquet (par exemple classer des paquets selon les champs qu'ils comportent).

Enfin, nous formalisons certains messages vers des structures de données afin de rendre l'analyse postérieure facile. Par exemple, certains messages de signalisation peuvent être lus et transformés en des structures de données de type graphe connecté afin de construire la topologie complète du réseau. Une fois que toutes les transformations sont faites, nous avons à notre disposition un Comma Separated Values (CSV) standardisé que nous allons pouvoir analyser en détail.

Méthodes & outils utilisés Dans le cas des fichiers PCAP, il n'existe pas pour le moment de bibliothèques en Python permettant de décoder tous les formats de paquets que nous utilisons. Cependant, il est possible de contrôler par Python l'appel à un outil extérieur. En l'occurrence, il s'agit de Tshark qui dispose d'un très large panel de décodeur pour protocoles parmi lesquels on trouve RPL et CoAP. Tshark a pour avantage de transformer une trace en un format texte qui offre une plus grande facilité de traitement. Un exemple de code est fourni en annexe A.3.

Justification de cette méthode Il serait possible d'utiliser des outils graphiques comme Wireshark pour pouvoir faire de l'exploration ad hoc. Bien que cette méthode soit utile durant les phases d'exploration et de consolidation du protocole expérimental, elle est lente lorsque le système est rodé. Les interfaces graphiques ne sont pas conçues pour des traitements massifs d'information. En outre, les changements de filtres dans Wireshark nécessitent de nouveaux calculs sur l'ensemble des fichiers capturés. À l'inverse une fois que les données essentielles sont mises dans un fichier texte, on se retrouve sur des données qu'il est facile de filtrer avec des outils classiques.

L'analyse de traces réseau permet d'analyser son comportement, d'identifier les noeuds actifs, d'identifier des problèmes éventuels et de collecter des statistiques diverses sur l'activité du réseau. La capture des paquets dans notre cas peut se faire au niveau de COOJA qui peut exporter des traces réseau ou bien au niveau d'un testbed qui est équipé de sniffer qui dupliquent tous les paquets reçus ou émis sur une interface.

Comparaisons avec les autres méthodes / Etat de l'art Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

5.2.4 Analyse des résultats

Description de l'étape Cette étape utilise les traces mises en formes obtenues à l'étape précédente. C'est dans cette étape que va se faire la recherche et la production de résultats qualitatifs et quantitatifs sur notre étude. Les calculs sur les graphes tels que la profondeur d'un nœud, la connexité du graphe ou bien la liste des plus courts chemins sont calculés. Enfin, au sujet du trafic réseau il est possible de fournir des statistiques sur le type de trafic émis, leur évolution dans le temps, le contenu des paquets. . .

Méthodes & outils utilisés Pandas [73] est une bibliothèque Python permettant la manipulation et l'analyse des données. Elle propose en particulier des structures de données et des opérations de manipulation de tableaux numériques complexes et de séries temporelles. Dans notre cas d'exploration du réseau, nous avons utilisé NetworkX. Cette bibliothèque nous permet de gérer les données liées à des graphes. Dans nos cas, nous voulions avoir une représentation du DODAG RPL, afin de calculer les plus courts chemins entre un nœud et la racine du DODAG. NetworkX nous apporte des facilités de calcul et de représentation graphique nous permettant de visualiser l'ensemble des liens pertinents dans le réseau. Des extraits de code sont disponibles en annexe A.4.

Justification de cette méthode Il serait possible d'utiliser des outils de filtres texte comme Awk ou sed ainsi que d'autres utilitaires UNIX classiques. Ils sont utiles dans des cas très simples, mais dès qu'il s'agit de grouper et de faire des agrégations complexes leur utilisation peut devenir difficile.

Dans le cas d'analyses sur de larges traces, il est opportun de découpler cette étape afin de la faire tourner sur un autre serveur disposant d'une puissance de calcul et d'une disponibilité supérieure à celle disponible sur sa propre machine. Le fait de pouvoir déplacer les traces et les traitements sur d'autres machines dynamiquement permet une plus grande agilité quand les expériences grandissent.

Comparaisons avec les autres méthodes / Etat de l'art Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

5.2.5 Présentation des résultats

Description de l'étape Cette ultime étape est celle où les résultats peuvent être visualisés et mis en forme. C'est également lors de cette phase que l'on va produire les graphes qui vont être utilisés dans une publication. L'utilisation d'un notebook aide cette consultation qui peut être faite dans un navigateur web. En outre, si l'on consulte ce notebook de manière interactive il est possible de ré exécuter certaines cellules sans connaître le détail de l'implémentation ce qui facilite les collaborations.

Méthodes & outils utilisés Jupyter peut convertir les notebooks en d'autres formats tels que HTML et PDF afin de faciliter l'impression et le partage de documents. En outre, le rendu graphique

est effectué par Matplotlib [50] qui est une bibliothèque de rendu graphique classique.

Justification de cette méthode L'intérêt d'utiliser ce genre de bibliothèque au lieu d'une solution spécialisée telle que Gnuplot [112] vient du simple fait qu'elle peut être intégrée directement à la suite de l'analyse et de nos traitements. En effet, il est possible dans pandas et dans NetworkX de chaîner des opérations et d'obtenir un graphe à la suite d'une série de filtres et d'agrégations. Un exemple de code est disponible en annexe A.5.

Justification de cette méthode Au lieu de créer un rapport qui soit un ensemble de fichiers disjoints ou bien d'un document statique, on peut créer un rapport interactif qui permette de ré-exécuter certaines portions de code. L'annotation et la modification des textes qui est en marge des résultats permettent d'avoir une façon beaucoup plus lisible de partager des résultats. Enfin grâce aux outils de conversions fournis, il est possible de transformer le notebook en un script unique qui peut être exécuté pour reproduire l'intégralité des résultats précédents. Ces outils de conversion mitigent la contrainte d'avoir un format de ce fichier commun à l'ensemble d'une équipe et permettent d'offrir un compromis intéressant en termes de choix communs et de fonctionnalités offertes.

Comparaisons avec les autres méthodes / Etat de l'art Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

5.2.6 Garanties de reproductibilité par intégration continue

Description de l'étape L'intégration continue est un ensemble de pratiques utilisées en production de logiciel qui consiste à vérifier que chaque modification apportée au code source ne modifie pas la correction du code et n'introduit pas de régression. Le principal but de cette pratique est de détecter les problèmes et les erreurs afin de les corriger au plus tôt.

Dans le cas d'expériences se déroulant sur un banc de test, il serait très complexe d'assurer la reproductibilité parfaite. En effet, l'environnement du banc de test est un système réel changeant. De plus, le matériel du banc de test peut être indisponible, remplacé, altéré.

Dans le cas de simulations, le but est d'assurer que l'environnement de simulation est reproductible. Ainsi, il est possible pour n'importe quel lecteur de reproduire les conditions dans lesquelles les expériences ont été exécutées et produites. Dans le cas d'expériences aléatoires comme les nôtres, un simulateur peut être complètement reproductible dès lors que la graine du générateur de nombres aléatoires a été fixée. Dès lors, à jeu de paramètres égaux les résultats sont constants.

Méthodes & Outils utilisés Si le notebook que nous avons utilisé jusqu'à présent est transformé en script, il suffit de l'exécuter pour reproduire l'intégralité de l'expérience. Nous avons lors de cette étude, hébergé notre code sur Github. Cette plateforme peut être couplée à de nombreux services d'intégration continue. Travis-ci [42] est l'un d'entre eux. Il permet de construire et d'exécuter une série de tests lorsqu'une modification proposée ou effective du code a lieu. Lorsque la série de tests est terminée, on peut ainsi vérifier qu'aucune régression des fonctionnalités n'a été commise. Ce service utilisé entre autres pour tester le système Contiki [31] peut l'être pour reproduire des expériences. Ainsi nous avons pu construire une démonstration présentant les différentes étapes présentées dans ce chapitre et les avons exécutées sur cette plateforme [66]. Le fait que Travis-ci soit une structure indépendante et ouverte renforce le fait que n'importe qui pourrait refaire l'expérience et s'assurer de nos résultats. Un extrait du fichier de configuration nécessaire pour le fonctionnement de cette plateforme peut être consulté en annexe A.6

Justification de cette méthode La possibilité de vérifier rapidement qu'une expérience a bel et bien fonctionné par un tiers de confiance est un bon gage sur le travail effectué. Une fois que cette vérification a été faite, il est aisé de trouver l'environnement logiciel qui a permis la production des

résultats pour être capable de partir de cet environnement afin de le modifier à nouveau et d'itérer rapidement afin de produire de nouveaux résultats.

On inscrit également les dépendances et on peut contrôler leur évolution au cours du temps afin d'avoir un environnement de test qui est contrôlé et versionné. Ainsi on a la garantie que la migration d'une dépendance vers une autre ne casse pas le flot de l'expérience ni ses résultats.

Comparaisons avec les autres méthodes / Etat de l'art Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

5.3 Conclusion & Perspectives

Problématique La reproductibilité d'une expérience est une question complexe, mais doit devenir une problématique de premier plan. À l'inverse d'autres disciplines scientifiques, l'informatique et le réseau jouissent d'une situation enviable où le coût pour reproduire une expérience est souvent modeste.

Outils De nombreux logiciels en licence libre ou bien des plateformes intégrées telles que Jenkins ou Travis-ci. Reconnaissance par la communauté : Des plateformes de partage de travaux scientifiques tel que HAL permettent d'héberger les fichiers nécessaires. La création de plateformes facilement automatisables telle qu'Iotlab va dans le bon sens. Il s'agit d'encourager les API ouvertes, documentées et stables plutôt que de vouloir un outil unique et spécialisée dans la résolution complète d'un problème.

Perspectives Les meilleures pratiques de développement logiciel influencent les méthodologies des chercheurs en informatique. La création de communautés autour des différents outils logiciels facilite les échanges et la diffusion de nouvelles méthodes. Cependant, l'investissement en temps ne peut être justifié que si la reproductibilité est une nécessité des projets de recherche. Encourager les tests et l'intégration continue au sein des projets de recherche. Même si de nombreux progrès ont été faits, il reste des chantiers ouverts. Dans le cas des réseaux de capteurs, l'intégration de nouvelles plateformes matérielles et des matrices de tests ajoutent aux défis logiciels ceux de l'émulation et des limites de celle-ci.

Conclusion

Comme nous l'avons vu tout au long de cette thèse, les passerelles peuvent et sont utilisés pour de multiples usages. En plus des fonctionnalités basiques de connexion et de sécurité. D'autres fonctions réseaux peuvent y être implémentée.

Les fonctionnalités de cache

Les fonctionnalités de supervision Dans le futur, il est possible que l'ensemble des fonctionnalités présentées soient présentes à un niveau complètement virtualisé sous la forme d'une Network Function Virtualization (NFV).

Les problématiques apportées par les LLN nous permettent d'étendre l'Internet a un plus large ensemble pour construire l'Internet du futur.

Codes Sources

A.1 Fabrication

```
with open(pj(path, "main.csc"), "w") as f:
    f.write(main_csc_template.render(
        title="Dummy Simulation",
        random_seed=12345,
        transmitting_range=42,
        interference_range=42,
        success_ratio_tx=1.0,
        success_ratio_rx=1.0,
        mote_types=[
            {"name": "server", "description": "server",
             "firmware": "dummy-server.wismote"},
            {"name": "client", "description": "client",
             "firmware": "dummy-client.wismote"}
        ],
        motes=[
            {"mote_id": 1, "x": 0, "y": 0, "z": 0, "mote_type": "server"},
            {"mote_id": 2, "x": 1, "y": 1, "z": 0, "mote_type": "client"},
        ],
        script=script))
```

```
<?xml version="1.0" encoding="UTF-8"?>
<simconf>
  <simulation>
    <title>{{ title }}</title>
    <randomseed>{{ random_seed }}</randomseed>
    <radiomedium>
      org.contikios.cooja.radiomediums.UDGM
    <transmitting_range>{{ transmitting_range }}</transmitting_range>
    <interference_range>{{ interference_range }}</interference_range>
    <success_ratio_tx>{{ success_ratio_tx }}</success_ratio_tx>
    <success_ratio_rx>{{ success_ratio_rx }}</success_ratio_rx>
```

```

</radiomedium>
{% for mote_type in mote_types %}
<motetype>
  org.contikios.cooja.mspmote.WismoteMoteType
  <identifier>{{ mote_type.name }}</identifier>
  <firmware EXPORT="copy">{{ mote_type.firmware }}</firmware>
</motetype>
{% endfor %}
{% for mote in motes %}
<mote>
  <interface_config>
    org.contikios.cooja.interfaces.Position
    <x>{{ mote.x }}</x>
    <y>{{ mote.y }}</y>
    <z>{{ mote.z }}</z>
  </interface_config>
  <interface_config>
    org.contikios.cooja.mspmote.interfaces.MspMoteID
    <id>{{ mote.mote_id }}</id>
  </interface_config>
  <motetype_identifier>{{ mote.mote_type }}</motetype_identifier>
</mote>
{% endfor %}
</simulation>
<plugin>
  org.contikios.cooja.plugins.ScriptRunner
  <plugin_config>
    <script>
      {{ script }}
    </script>
  </plugin_config>
</plugin>
</simconf>

```

Nous pouvons voir des cas de paramètres simples avec un remplacement clé → valeur.

Il est également possible d'avoir des boucles, des itérations et des conditionnelles. C'est le cas par exemple dans la création des emplacements des noeuds introduits grâce à la boucle `for`.

Notons ici que la balise `script` utilise également un moteur de templating. Ainsi, la configuration du programme qui peut lui aussi contenir des conditionnels et des boucles d'exécution peut être aussi géré avec ce mécanisme.

A.2 Déploiement

```
import fabric

@host("grenoble")
def run():
    run_experiment
```

A.3 Parsing

```
import subprocess
from os.path import join as pj

def pcap2csv(folder, filename="output.csv"):
    """
    Execute a simple filter on PCAP and count
    """
    # Getting raw data
    with open(pj(folder, filename), "w") as f:

        command = ["tshark",
                    "-T", "fields",
                    "-E", "header=y",
                    "-E", "separator=",
                    "-Y", "udp || icmpv6",
                    "-e", "frame.time_epoch",
                    "-e", "frame.protocols",
                    "-e", "frame.len",
                    "-e", "wpan.fcs",
                    "-e", "wpan.seq_no",
                    "-e", "wpan.src16",
                    "-e", "wpan.dst16",
                    "-e", "wpan.src64",
                    "-e", "wpan.dst64",
                    "-e", "icmpv6.type",
                    "-e", "ipv6.src",
                    "-e", "ipv6.dst",
                    "-e", "icmpv6.code",
                    "-e", "udp.dstport",
                    "-e", "udp.srcport",
                    "-e", "data.data",
                    "-r", pj(folder, "output.pcap")]

        process = subprocess.Popen(command, stdout=subprocess.PIPE)
        stdout, stderr = process.communicate()
        f.write(stdout)
```

Afin d'avoir un traitement des données aisé, il est préférable de se ramener à des formats de fichiers textuels tel que le CSV. Ici, tshark est utilisé comme intermédiaire pour traduire un format binaire et extraire les informations les plus essentielles vers du texte.

Il est à noter que dès que cette transformation a été faite une fois, elle n'est jamais effectuée. En effet, le fait d'avoir sauvegardé les données dans un format intermédiaire permet de ne travailler qu'avec le CSV et de ne plus jamais avoir à faire une extraction d'information coûteuse et redondante. De plus dans le cas d'un fichier binaire au format changeant, avoir une version texte garantis que ces données pourront être lus postérieurement si le traducteur (en l'occurrence tshark) n'est plus disponible.

A.4 Analyse

```
import pandas as pd

df = pd.read_csv("my_results.csv")
df[df.pkt_type == "udp"].count()
```

A.4.1 Filtrage

Nous obtenons ainsi en une ligne de code, un graphe représentant l'histogramme du nombre de paquets.

A.4.2 Fonctions agrégées

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

A.4.3 Traitements en masse

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

A.5 Présentation

```
import matplotlib.pyplot as plt
import pandas as pd

df = pd.read_csv("my_results.csv")
df[df.pkt_type == "udp"].count().plot(kind="bar")
```

A.6 Intégration continue (Travis-ci)

```
language: python
```

```
# Mise en place de l'environnement
```

```
install:
```

```
    # La compilation des sources de la suite python peut être un
    # peu longue. Il est commode d'utiliser des paquets binaires afin
    # d'avoir un build rapide
    - "sudo apt-get -qq install ipython"
    - "sudo apt-get -qq install python-matplotlib"
    - "sudo apt-get -qq install fabric"
    - "sudo apt-get -qq install python-pandas"
    - "sudo apt-get -qq install python-networkx"
    - "sudo apt-get -qq install python-numpy"
    - "sudo apt-get -qq install python-jinja2"
    - "sudo apt-get -qq install python-scipy"

    # Récupération des sources de contiki et des compilateurs nécessaires
    - "git clone --recursive git://github.com/contiki-os/contiki"
    - sudo apt-get -qq update
    - sudo apt-get -qq install lib32z1
    - wget http://simonduq.github.io/resources/mspgcc-4.7.2-compiled.tar.bz2 &&
      tar xjf mspgcc*.tar.bz2 -C /tmp/ &&
      sudo cp -f -r /tmp/msp430/* /usr/local/ &&
      rm -rf /tmp/msp430 mspgcc*.tar.bz2 &&
      msp430-gcc --version

    # Utile pour analyser les traces PCAP
    - "sudo apt-get install tshark"
```

```
# Execution. Si ces commandes renvoient 0
```

```
# le test est considéré comme un succès
```

```
script:
```

```
    - pip install -r requirements.txt
    # Conversion du notebook vers un script exécutable
    - "ipython nbconvert --to=python demo.ipynb"
    # Execution du script
    - python demo.py
```

L'utilisation de Travis-ci peut être justifié par le fait qu'un tiers à priori non lié à une organisation ou un organisme de recherche peut être de bonne foi quant au caractère reproductible. L'intégration continue garantit que l'expérience pourra être reproduite. Un point important de cette configuration est que les logiciels sont spécifiés avec une version (requirements.txt). Ainsi, au cas où la bibliothèque viendrait à ne plus être maintenue ou comporterait des changements d'interfaces, il serait toujours possible de retrouver d'anciennes versions et reproduire l'expérience avant de la faire migrer vers de nouvelles versions.

Acronymes

h Cache Hit ratio
m Cache Miss ratio
6LBR 6LoWPAN Border Router
ACK Acknowledgement message
API Application Programming Interface
BAN Body-Area Network
CBOR Concise Binary Object Representation
CoAP Constrained Application Protocol
CON Confirmable message
CoRE Constrained RESTful environment
CSMA/CA Carrier Sense Multiple Access with Collision Avoidance
CSV Comma Separated Values
DAD Duplicate Address Detection
DAG Directed Acyclic Graph
DAO Destination Advertisement Object
DIO DODAG Information Object
DIS DODAG Informational Solicitation
DODAG Destination-Oriented DAG
DSL Domain Specific Language
DTLS Datagram Transport Layer Security
EWMA Exponentially Weighted Moving Average
FFD Full Function Device
HCP HTTP-CoAP proxy
HTTP HyperText Transfer Protocol
ICMPv6 Internet Control Message Protocol v6
IETF Internet Engineering Task Force
IHM Interaction Homme Machine

IID Interface ID
IoT Internet of Things
IP Internet Protocol
JSON Javascript Serial Object Notation
LAN Local Area Network
LBR LoWPAN Border Router
LLN Low-Power and Lossy Networks
LoWPAN Low-Power Wireless Personal Area Network
LRWPAN Low Rate Wireless Personal Area Network
M2M Machine to Machine
MAC Media access control
MP2P Multi-point to point
MTU Maximum transmission unit
NAT Network Address Translation
NDP Neighbor Discovery Protocol
NFV Network Function Virtualization
NON Non-confirmable message
NSGA Non-dominated Sorting Genetic Algorithm
OTA Over The Air
P2MP Point to Multi-point
P2P Point to Point
PCAP Packet CAPture
QoS Quality of Service
REST REpresentational State Transfer
RFD Reduced Function Device
ROLL Routing Over Low power and Lossy Networks
RPL Routing Protocol Layer
RST Reset message
SCADA Supervisory Control and Data Acquisition
SOA Service Oriented Architecturei
SQL Structured Query Language
TCP Transport Control Protocol
TDMA Time Division Multiple Access
Tee Traffic Energy Estimator
TTL Time To Live
UDP User Datagram Protocol
URI Uniform Resource Identifier
URL Uniform Resource Locator
WSN Wireless Sensor Networks
XML Extensible Markup Language

Bibliographie

- [1] 6tisch. IPv6 over the TSCH mode of IEEE 802.15.4e. <http://datatracker.ietf.org/wg/6tisch/>.
- [2] Cesare Alippi, Giuseppe Anastasi, Cristian Galperti, Francesca Mancini, and Manuel Rove. Adaptive sampling for energy conservation in wireless sensor networks for snow monitoring applications. In *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE International Conference on*, pages 1–6. IEEE, 2007.
- [3] ZigBee Alliance. Zigbee specification, 2006.
- [4] Carles Anton-Haro and Mischa Dohler. *Machine-to-machine (M2M) Communications : Architecture, Performance and Applications*. Elsevier, 2014.
- [5] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things : A survey. *Computer networks*, 54(15) :2787–2805, 2010.
- [6] Emmanuel Baccelli, Oliver Hahm, Mesut Gunes, Matthias Wahlisch, and Thomas C Schmidt. Riot os : Towards an os for the internet of things. In *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*, pages 79–80. IEEE, 2013.
- [7] Nouha Baccour, Anis Koubaa, Luca Mottola, Marco Antonio Zuniga, Habib Youssef, Carlo Alberto Boano, and Mário Alves. Radio link quality estimation in wireless sensor networks : a survey. *ACM Transactions on Sensor Networks (TOSN)*, 8(4) :34, 2012.
- [8] Debasis Bandyopadhyay and Jaydip Sen. Internet of things : Applications and challenges in technology and standardization. *Wireless Personal Communications*, 58(1) :49–69, 2011.
- [9] Paolo Baronti, Prashant Pillai, Vince WC Chook, Stefano Chessa, Alberto Gotta, and Y Fun Hu. Wireless sensor networks : A survey on the state of the art and the 802.15. 4 and zigbee standards. *Computer communications*, 30(7) :1655–1695, 2007.
- [10] Olaf Bergmann. libcoap : C-implementation of coap. <http://libcoap.net>, 2012.
- [11] C. Bormann and P. Hoffman. Concise Binary Object Representation (CBOR). RFC 7049 (Proposed Standard), October 2013.
- [12] S Brown and CJ Sreenan. Updating software in wireless sensor networks : A survey. *Dept. of Computer Science, National Univ. of Ireland, Maynooth, Tech. Rep*, 2006.
- [13] Tomasz Buchert, Cristian Ruiz, Lucas Nussbaum, and Olivier Richard. A survey of general-purpose experiment management tools for distributed systems. *Future Generation Computer Systems*, 45 :1–12, 2015.

-
- [14] Qing Cao, Ting Yan, John Stankovic, and Tarek Abdelzaher. Analysis of target detection performance for wireless sensor networks. In *Distributed Computing in Sensor Systems*, pages 276–292. Springer, 2005.
 - [15] Andrea Caragliu, Chiara Del Bo, and Peter Nijkamp. Smart cities in europe. *Journal of urban technology*, 18(2) :65–82, 2011.
 - [16] A. Castellani. Guidelines for HTTP-CoAP Mapping Implementations. Internet-Draft draft-ietf-core-http-mapping-07, Internet Engineering Task Force. Work in progress.
 - [17] Marco Centenaro, Lorenzo Vangelista, Andrea Zanella, and Michele Zorzi. Long-range communications in unlicensed bands : the rising stars in the iot and smart city scenarios. *arXiv preprint arXiv :1510.00620*, 2015.
 - [18] Edward Chan and Song Han. Energy efficient residual energy monitoring in wireless sensor networks. *International Journal of Distributed Sensor Networks*, 5(6), 2009.
 - [19] Bor-rong Chen, Geoffrey Peterson, Geoff Mainland, and Matt Welsh. Livenet : Using passive monitoring to reconstruct sensor network dynamics. In *IEEE/ACM DCOSS*, 2008.
 - [20] Min Chen, Sergio Gonzalez, Athanasios Vasilakos, Huasong Cao, and Victor C Leung. Body area networks : A survey. *Mobile networks and applications*, 16(2) :171–193, 2011.
 - [21] Chipcon. *2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver*.
 - [22] Simone Cirani, Luca Davoli, Gianluigi Ferrari, Rémy Léone, Paolo Medagliani, Marco Picone, and Luca Veltri. A scalable and self-configuring architecture for service discovery in the internet of things. 2014.
 - [23] Thomas Clausen, Ulrich Herberg, and Matthias Philipp. A critical evaluation of the ipv6 routing protocol for low power and lossy networks (rpl). In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2011 IEEE 7th International Conference on*, pages 365–372. IEEE, 2011.
 - [24] W. Colitti, K. Steenhaut, and N. De Caro. Integrating wireless sensor networks with the web. *Extending the Internet to Low power and Lossy Networks (IP+ SN 2011)*, 2011.
 - [25] Walter Colitti, Kris Steenhaut, Niccolo De Caro, Bogdan Buta, and Virgil Dobrota. Rest enabled wireless sensor networks for seamless integration with web applications. In *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on*, pages 867–872. IEEE, 2011.
 - [26] Moteiv Corp. Ultra low power IEEE 802.15.4 compliant wireless sensor module.
 - [27] Bart De Schutter and Bart De Moor. Optimal traffic light control for a single intersection. *European Journal of Control*, 4(3) :260–276, 1998.
 - [28] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm : NSGA-II. *IEEE Trans. on Evolutionary Computation*, 6(2) :182–197, April 2002.
 - [29] Kalyanmoy Deb and Samir Agrawal. A niched-penalty approach for constraint handling in genetic algorithms. In *Artificial Neural Nets and Genetic Algorithms*, pages 235–243. Springer, 1999.
 - [30] A. Dunkels, B. Gronvall, and T. Voigt. Contiki-a lightweight and flexible operating system for tiny networked sensors. pages 455–462, 2004.
 - [31] Adam Dunkels. Contiki regression tests : 9 hardware platforms, 4 processor architectures, 1021 network nodes. <http://contiki-os.blogspot.fr/2012/12/contiki-regression-tests-9-hardware.html>.
 - [32] Adam Dunkels. The ContikiMAC Radio Duty Cycling Protocol. Technical Report T2011 :13, Swedish Institute of Computer Science, December 2011.

-
- [33] Adam Dunkels, Joakim Eriksson, Niclas Finne, and Nicolas Tsiftes. Powertrace : Network-level power profiling for low-power wireless networks. 2011.
 - [34] Simon Duquennoy, Niklas Wiström, Nicolas Tsiftes, and Adam Dunkels. Leveraging ip for sensor network deployment. In *Proceedings of the workshop on Extending the Internet to Low power and Lossy Networks (IP+ SN 2011)*, Chicago, IL, USA, volume 11. Citeseer, 2011.
 - [35] Gregory A Ehlers, Robert D Howerton, and Gary E Speegle. Engery management and building automation system, November 5 1996. US Patent 5,572,438.
 - [36] Joakim Eriksson, Adam Dunkels, Niclas Finne, Fredrik Osterlind, and Thiemo Voigt. Mspsim— an extensible simulator for msp430-equipped sensor boards. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session*, page 27, 2007.
 - [37] Melike Erol-Kantarci and Hussein T Mouftah. Wireless sensor networks for cost-efficient residential energy management in the smart grid. *Smart Grid, IEEE Transactions on*, 2(2) :314–325, 2011.
 - [38] Konstantinos P Ferentinos and Theodore A Tsiligiridis. Adaptive design optimization of wireless sensor networks using genetic algorithms. *Computer Networks*, 51(4) :1031–1051, 2007.
 - [39] Sir Ronald Aylmer Fisher, Statistiker Genetiker, Ronald Aylmer Fisher, Statistician Genetician, Ronald Aylmer Fisher, and Statisticien Généticien. *The design of experiments*, volume 12. Oliver and Boyd Edinburgh, 1960.
 - [40] Eric Fleury, Nathalie Mitton, Thomas Noel, Cédric Adjih, Valeria Loscri, Anna Maria Vegni, Riccardo Petrolo, Valeria Loscri, Nathalie Mitton, Gianluca Aloï, et al. Fit iot-lab : The largest iot open experimental testbed. *ERCIM News*, (101) :14, 2015.
 - [41] Jeff Forcier. Fabric pythonic remote execution. <http://www.fabfile.org>.
 - [42] Travis CI GmbH. Travis CI continuous integration and deployment that just works. <https://travis-ci.com/>.
 - [43] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot) : A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7) :1645–1660, 2013.
 - [44] Dominique Guinard, Vlad Trifa, and Erik Wilde. A resource oriented architecture for the web of things. In *Internet of Things (IOT), 2010*, pages 1–8. IEEE, 2010.
 - [45] Vehbi C Gungor, Bin Lu, and Gerhard P Hancke. Opportunities and challenges of wireless sensor networks in smart grid. *Industrial Electronics, IEEE Transactions on*, 57(10) :3557–3564, 2010.
 - [46] Jane K Hart and Kirk Martinez. Environmental sensor networks : A revolution in the earth system science ? *Earth-Science Reviews*, 78(3) :177–191, 2006.
 - [47] Jason Hill, Mike Horton, Ralph Kling, and Lakshman Krishnamurthy. The platforms enabling wireless sensor networks. *Communications of the ACM*, 47(6) :41–46, 2004.
 - [48] Robert G Hollands. Will the real smart city please stand up ? intelligent, progressive or entrepreneurial ? *City*, 12(3) :303–320, 2008.
 - [49] Peng Hu, Zude Zhou, Quan Liu, and Fangmin Li. The hmm-based modeling for the energy level prediction in wireless sensor networks. In *IEEE ICIEA*, Harbin, China, 2007.
 - [50] J. D. Hunter. Matplotlib : A 2d graphics environment. *Computing In Science & Engineering*, 9(3) :90–95, 2007.
 - [51] Philipp Hurni, Benjamin Nyffenegger, Torsten Braun, and Anton Hergenroeder. On the accuracy of software-based energy estimation techniques. In *Wireless Sensor Networks*, pages 49–64. Springer, 2011.

-
- [52] Damien B Jourdan and Olivier L de Weck. Multi-objective genetic algorithm for the automated planning of a wireless sensor network to monitor a critical facility. In *Defense and Security*, pages 565–575. International Society for Optics and Photonics, 2004.
 - [53] Simon Kellner, Mario Pink, Detlev Meier, and E-O Blass. Towards a realistic energy model for wireless sensor networks. In *Wireless on Demand Network Systems and Services, 2008. WONS 2008. Fifth Annual Conference on*, pages 97–100. IEEE, 2008.
 - [54] Kavi K Khedo, Rajiv Perseedoss, Avinash Mungur, et al. A wireless sensor network air pollution monitoring system. *arXiv preprint arXiv :1005.1737*, 2010.
 - [55] Sukun Kim, Shamim Pakzad, David Culler, James Demmel, Gregory Fenves, Steven Glaser, and Martin Turon. Health monitoring of civil infrastructures using wireless sensor networks. In *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, pages 254–263. IEEE, 2007.
 - [56] Jonathan G Koomey, Stephen Berard, Marla Sanchez, and Henry Wong. Implications of historical trends in the electrical efficiency of computing. *Annals of the History of Computing, IEEE*, 33(3) :46–54, 2011.
 - [57] Matthias Kovatsch, Simon Duquennoy, and Adam Dunkels. A low-power coap for contiki. In *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on*, pages 855–860. IEEE, 2011.
 - [58] Matthias Kovatsch, Martin Lanter, and Zach Shelby. Californium : Scalable cloud services for the internet of things with coap. In *Internet of Things (IOT), 2014 International Conference on the*, pages 1–6. IEEE, 2014.
 - [59] Matthias Kovatsch, Simon Mayer, and Benedikt Ostermaier. Moving Application Logic from the Firmware to the Cloud : Towards the Thin Server Architecture for the Internet of Things. In *Proc of the 6th Int Conf on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS 2012), Palermo, Italy, July 2012*.
 - [60] N. Kushalnagar, G. Montenegro, and C. Schumacher. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs) : Overview, Assumptions, Problem Statement, and Goals. RFC 4919 (Informational), August 2007.
 - [61] Mathieu Lacage, Martin Ferrari, Mads Hansen, Thierry Turletti, and Walid Dabbous. Nepi : using independent simulators, emulators, and testbeds for easy experimentation. *ACM SIGOPS Operating Systems Review*, 43(4) :60–65, 2010.
 - [62] Abdelkader Lahmadi, Alexandre Boeglin, and Olivier Festor. Efficient distributed monitoring in 6lowpan networks. In *CNSM, Zurich, Switzerland, 2013*.
 - [63] Koen Langendoen. Medium access control in wireless sensor networks. *Medium access control in wireless networks*, 2 :535–560, 2008.
 - [64] Kevin C Lee, Raghuram Sudhaakar, Lillian Dai, Sateesh Addepalli, and Mario Gerla. Rpl under mobility. In *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*, pages 300–304. IEEE, 2012.
 - [65] Tomas Lennvall, Stefan Svensson, and Fredrik Hekland. A comparison of wireless hART and zigbee for industrial applications. In *IEEE International Workshop on Factory Communication Systems*, volume 2008, pages 85–88, 2008.
 - [66] Rémy Léone, Jérémie Leguay, Paolo Medagliani, and Claude Chaudet. Demo abstract : Makesense—managing reproducible wsns experiments. In *Real-World Wireless Sensor Networks*, pages 65–71. Springer, 2014.
 - [67] Rémy Leone, Jeremie Leguay, Paolo Medagliani, Claude Chaudet, et al. Makesense : Managing reproducible wsns experiments. *Fifth Workshop on Real-World Wireless Sensor Networks*, 2013.

-
- [68] Rémy Leone, Paolo Medagliani, and Jérémie Leguay. Optimizing QoS in Wireless Sensors Networks using a Caching Platform. In *Sensornets 2013*, page 56, Barcelone, Espagne, February 2013.
- [69] Slawek Ligus. *Effective Monitoring and Alerting*. " O'Reilly Media, Inc.", 2012.
- [70] Changlei Liu and Guohong Cao. Distributed monitoring and aggregation in wireless sensor networks. In *IEEE INFOCOM*, San Diego, CA, USA, 2010.
- [71] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97. ACM, 2002.
- [72] Jerry Martocci, Pieter Mil, Nicolas Riou, and Wouter Vermeulen. Building automation routing requirements in low-power and lossy networks. 2010.
- [73] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.
- [74] Paolo Medagliani, Jérémie Leguay, Andrzej Duda, Franck Rousseau, Marc Domingo, Mischa Dohler, Ignasi Vilajosana, and Olivier Dupont. Bringing ip to low-power smart objects : the smart parking case in the calipso project.
- [75] Raquel A.F. Mini, Antonio A.F. Loureiro, and Badri Nath. The distinctive design characteristic of a wireless sensor network : the energy map. *Computer Communications*, 27, 2004.
- [76] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944 (Proposed Standard), September 2007. Updated by RFC 6282.
- [77] David Moss and Philip Levis. Box-macs : Exploiting physical and link layer boundaries in low-power networking. *Computer Systems Laboratory Stanford University*, pages 116–119, 2008.
- [78] San Murugesan. Harnessing green it : Principles and practices. *IT professional*, 10(1) :24–33, 2008.
- [79] B O'Flyrm, Ricardo Martinez, John Cleary, Catherine Slater, F Regan, Dermot Diamond, and H Murphy. Smartcoast : a wireless sensor network for water quality monitoring. In *Local Computer Networks, 2007. LCN 2007. 32nd IEEE Conference on*, pages 815–816. Ieee, 2007.
- [80] Luís ML Oliveira, Amaro F De Sousa, and Joel JPC Rodrigues. Routing and mobility approaches in ipv6 over lowpan mesh networks. *International Journal of Communication Systems*, 24(11) :1445–1466, 2011.
- [81] Fredrik Österlind, Adam Dunkels, Joakim Eriksson, Niclas Finne, and Thiemo Voigt. Cross-level sensor network simulation with cooja. In *LCN*, 2006.
- [82] Fredrik Österlind, Joakim Eriksson, and Adam Dunkels. Cooja timeline : a power visualizer for sensor network simulation. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 385–386. ACM, 2010.
- [83] Tae Rim Park, Tae Hyun Kim, Jae Young Choi, Sunghyun Choi, and Wook Hyun Kwon. Throughput and energy consumption analysis of ieee 802.15. 4 slotted csma/ca. *Electronics Letters*, 41(18) :1017–1019, 2005.
- [84] Al-Sakib Khan Pathan, Hyung-Woo Lee, and Choong Seon Hong. Security in wireless sensor networks : issues and challenges. In *Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference*, volume 2, pages 6–pp. IEEE, 2006.
- [85] Roger D Peng. Reproducible research in computational science. *Science (New York, Ny)*, 334(6060) :1226, 2011.

-
- [86] Fernando Pérez and Brian E. Granger. IPython : a system for interactive scientific computing. *Computing in Science and Engineering*, 9(3) :21–29, May 2007.
- [87] Pocoo. Jinja2 jinja2 is a templating engine for python. <http://jinja.pocoo.org>.
- [88] J. Polastre, R. Szewczyk, and D. Culler. Telos : enabling ultra-low power wireless research. In *Proc. of the 4th Int. Symp. on Information Processing in Sensor Networks (IPSN 05)*, pages 364 – 369, Piscataway, NJ, 2005.
- [89] Karl Raimund Popper. *The Logic of Scientific Discovery*. Routledge, 2002. 1st English Edition :1959.
- [90] Narendra PrithviRaj, Simon Duquennoy, and Thiemo Voigt. Ble and ieee 802.15. 4 in the iot : Evaluation and interoperability considerations. In *International Conference on Interoperability in IoT*, 2015.
- [91] Guy Pujolle. *Les réseaux : Edition 2014*. Editions Eyrolles, 2014.
- [92] Vijay Raghunathan, Aman Kansal, Jason Hsu, Jonathan Friedman, and Mani Srivastava. Design considerations for solar energy harvesting wireless embedded systems. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, page 64. IEEE Press, 2005.
- [93] David R Raymond and Scott F Midkiff. Denial-of-service in wireless sensor networks : Attacks and defenses. *Pervasive Computing, IEEE*, 7(1) :74–81, 2008.
- [94] Leonard Richardson and Sam Ruby. *RESTful web services*. " O'Reilly Media, Inc.", 2008.
- [95] Luis Ruiz-Garcia, Loredana Lunadei, Pilar Barreiro, and Ignacio Robla. A review of wireless sensor technologies and applications in agriculture and food industry : state of the art and current trends. *Sensors*, 9(6) :4728–4750, 2009.
- [96] Z. Shelby, S. Chakrabarti, E. Nordmark, and C. Bormann. Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs). RFC 6775 (Proposed Standard), November 2012.
- [97] Z. Shelby, K. Hartke, and C. Bormann. The Constrained Application Protocol (CoAP). RFC 7252 (Proposed Standard), June 2014.
- [98] Zach Shelby and Carsten Bormann. *6LoWPAN : The wireless embedded Internet*, volume 43. John Wiley & Sons, 2011.
- [99] Zhenyu Song, Mihai T Lazarescu, Riccardo Tomasi, Luciano Lavagno, and Maurizio A Spirito. High-level internet of things applications development using wireless sensor networks. In *Internet of Things*, pages 75–109. Springer, 2014.
- [100] Thanos Stathopoulos, John Heidemann, and Deborah Estrin. A remote code update mechanism for wireless sensor networks. Technical report, DTIC Document, 2003.
- [101] Andreas Terzis, Annalingam Anandarajah, Kevin Moore, I Wang, et al. Slip surface localization in wireless sensor networks for landslide prediction. In *Proceedings of the 5th international conference on Information processing in sensor networks*, pages 109–116. ACM, 2006.
- [102] Joydeep Tripathi, Jaudelice Cavalcante de Oliveira, and Jean-Philippe Vasseur. A performance evaluation study of rpl : Routing protocol for low power and lossy networks. In *Information Sciences and Systems (CISS), 2010 44th Annual Conference on*, pages 1–6. IEEE, 2010.
- [103] Nicolas Tsiftes, Joakim Eriksson, and Adam Dunkels. Low-power wireless ipv6 routing with contikirpl. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 406–407. ACM, 2010.
- [104] Malisa Vucinic, Bernard Tourancheau, and Andrzej Duda. Performance Comparison of the RPL and LOADng Routing Protocols in a Home Automation Scenario. In *IEEE WCNC*, 2013.

-
- [105] Klaus-Dieter Walter. Implementing m2m applications via gprs, edge and umts. *White paper—M2M Alliance*, 2009.
- [106] Ning Wang, Naiqian Zhang, and Maohua Wang. Wireless sensors in agriculture and food industry—recent development and future perspective. *Computers and electronics in agriculture*, 50(1) :1–14, 2006.
- [107] Yong Wang, Garhan Attebury, and Byrav Ramamurthy. A survey of security issues in wireless sensor networks. 2006.
- [108] Tim Wark, Peter Corke, Pavan Sikka, Lasse Klingbeil, Ying Guo, Chris Crossman, Phil Valencia, Dave Swain, and Greg Bishop-Hurley. Transforming agriculture through pervasive wireless sensor networks. *Pervasive Computing, IEEE*, 6(2) :50–57, 2007.
- [109] Thomas Watteyne, Xavier Vilajosana, Branko Kerkez, Fabien Chraim, Kevin Weekly, Qin Wang, Steven Glaser, and Kris Pister. Openwsn : a standards-based low-power wireless development environment. *Transactions on Emerging Telecommunications Technologies*, 23(5) :480–493, 2012.
- [110] Geoffrey Werner-Allen, Konrad Lorincz, Mario Ruiz, Omar Marcillo, Jeff Johnson, Jonathan Lees, and Matt Welsh. Deploying a wireless sensor network on an active volcano. *Internet Computing, IEEE*, 10(2) :18–25, 2006.
- [111] Duane Wessels. *Web caching*. " O'Reilly Media, Inc.", 2001.
- [112] Thomas Williams, Colin Kelley, and many others. Gnuplot 4.4 : an interactive plotting program. <http://gnuplot.info/>, March 2010.
- [113] Greg Wilson, DA Aruliah, C Titus Brown, Neil P Chue Hong, Matt Davis, Richard T Guy, Steven HD Haddock, Katy Huff, Ian M Mitchell, Mark D Plumbley, et al. Best practices for scientific computing. *PLoS biology*, 12(1) :e1001745, 2014.
- [114] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, JP Vasseur, and R. Alexander. RPL : IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550 (Proposed Standard), March 2012.
- [115] Kehui Xiao, Deqin Xiao, and Xiwen Luo. Smart water-saving irrigation system in precision agriculture based on wireless sensor network. *Transactions of the Chinese Society of Agricultural Engineering*, 26(11) :170–175, 2010.
- [116] Ye Yan, Yi Qian, Hamid Sharif, and David Tipper. A survey on smart grid communication infrastructures : Motivations, requirements and challenges. *Communications Surveys & Tutorials, IEEE*, 15(1) :5–20, 2013.
- [117] Liyang Yu, Neng Wang, and Xiaoqiao Meng. Real-time forest fire detection with wireless sensor networks. In *Wireless Communications, Networking and Mobile Computing, 2005. Proceedings. 2005 International Conference on*, volume 2, pages 1214–1217. IEEE, 2005.
- [118] Jerry Zhao, Ramesh Govindan, and Deborah Estrin. Residual energy scans for monitoring wireless sensor networks. In *IEEE WCNC*, 2002.
- [119] Qian Zhu, Ruicong Wang, Qi Chen, Yan Liu, and Weijun Qin. Iot gateway : Bridging wireless sensor networks into internet of things. In *Embedded and Ubiquitous Computing (EUC), 2010 IEEE/IFIP 8th International Conference on*, pages 347–352. IEEE, 2010.

Passerelles intelligentes pour réseaux de capteurs

Remy Leone

RESUME :

MOTS-CLEFS : bla bla bla

ABSTRACT :

KEY-WORDS : bla bla bla



