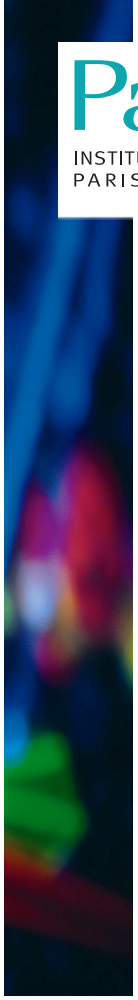


ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2016-ENST-0038

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

Rémy LEONE

6 novembre 2016

Passerelle intelligente pour réseaux de capteurs sans fil contraints

Directeur de thèse : **M. Jean Louis ROUGIER**
Co-encadrement de la thèse : **M. Vania CONAN**

Jury

M. Andrzej DUDA, Professeur, Grenoble INP, France	Rapporteur
Mme. Nathalie MITTON, Directrice de recherche, Inria Lille, France	Rapporteur
M. Fabrice THEOLEYRE, Chargé de recherche, Université de Strasbourg, France	Examinateur
M. Thomas WATTEYNE, Chargé de recherche, Inria Paris, France	Examinateur
M. Marcelo DIAS DE AMORIM, Directeur de recherche, UPMC, France	Examinateur
M. Jacques TIBERGHIEN, Professeur, Vrije Universiteit Brussel, Belgique	Examinateur
M. Jean Louis ROUGIER, Professeur, Télécom ParisTech, France	Directeur de thèse
M. Vania CONAN, Responsable de service, Thales Communications & Security, France	Directeur de thèse
M. Jeremie LEGUAY, Chef d'équipe, Huawei Technologies Co. Ltd, France	Encadrant
M. Claude CHAUDET, Maître de Conférences, Télécom ParisTech, France	Encadrant

T H È S E

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 Network (LLN)s are constrained networks composed by nodes with little resources (memory, CPU, battery...). Those networks are typically used in the context of Internet of Things (IoT) to provide real-time measurement of their environment. They got heterogeneous sensors and are used in various contexts such as home automation or smart cities. Gateways are used to provide a connectivity between such networks and standard IP ones. Due to its key location, the gateway has an accurate knowledge of information coming in and out of the LLN. This thesis offers three contributions dealing with optimization of resources and enhancing knowledge about its behavior.

The first contribution of this thesis is a method to implicitly measure node radio usage in order to infer their energy consumption. LLN's administrators need to know the state of the node which are relied upon for a given application to insure that it behaves correctly. But it needs to get the required information by spending as little as possible to get them. Explicit measurements of those information aren't always available and even so, they are costly in large scale LLNs. By monitoring the quantity of data exchanged with a device and knowing its nominal power consumption, we show that a gateway can estimate the state of the device's battery. It thus becomes possible to have an idea of the lifetime of each device without sending control messages costly in terms of power and bandwidth. Limit of this approach will be also described and a bias correction method will be offered when explicit measurements are possible.

Second we offer to determine the validity time of a response used within an applicative cache. Caches speed up request treatment and reduce the load over a LLN to increase its lifetime. Finding validity time for each request response must take into account multiple parameters and related works don't provide explicit methods to find the right time for a given configuration. We propose a method based on multi-objective optimization model to find the set of solution that are acceptable for our problem.

Finally, we needed a way to have reproducible experiment for each of our experiment. Therefore we provide Makesense, a framework that can document, execute and analyze a complete LLN experiment on simulation or real nodes from a unique description. We can combine fast development phases with many iterations then deploy the same code on real nodes and this performs realistic tests. Finally this framework can be coupled with automated testing to guarantee repeatable experiments and therefore enhance its usage by the scientific community.

Résumé

Les réseaux de capteurs (aussi appelés Low-Power and Lossy Network (LLN)s en anglais) sont des réseaux contraints composés de nœuds ayant de faibles ressources (mémoire, CPU, batterie). Leur utilisation est croissante dans le contexte de l'Internet of Things (IoT) afin d'obtenir des mesures en temps réel de l'environnement dans lequel ils sont déployés. Ils sont de nature très hétérogène et utilisés dans des contextes variés comme la domotique ou les villes intelligentes. Pour se connecter nativement à l'Internet, un LLN utilise une passerelle, qui du fait de sa position a une vue précise du LLN et des informations qui y rentrent et en sortent. Le but de cette thèse est d'exposer comment des fonctionnalités peuvent être ajoutées à la passerelle d'un LLN dans le but d'optimiser l'utilisation des ressources limitées du LLN et d'améliorer la connaissance de son état de fonctionnement.

La première contribution de cette thèse est une méthode de mesure implicite de l'utilisation de la radio des nœuds d'un LLN permettant d'inférer leur consommation énergétique. Pour s'assurer que le fonctionnement d'une application est correct, un administrateur d'un LLN a besoin de connaître l'état des nœuds dont dépend son application tout en dépensant aussi peu d'énergie que possible pour obtenir cette information. Mesurer explicitement ces informations n'est pas toujours possible et même quand c'est le cas, il est coûteux de le demander à chaque nœud dans un réseau de grande taille. En mesurant la quantité de données échangées avec un nœud et connaissant la consommation énergétique nominale de la radio nous montrons que la passerelle peut estimer l'énergie consommée par le nœud sans avoir à utiliser des messages de contrôle coûteux en terme de bande passante et de consommation énergétique. Les limites de cette approche seront également décrites et une correction des biais sera proposée lorsque des mesures explicites sont possibles.

La seconde contribution de cette thèse propose de déterminer les temps de validité de ressource optimaux à utiliser au sein d'un cache applicatif. L'utilisation d'un cache applicatif permet d'accélérer le traitement des requêtes et de réduire la charge d'un LLN dans le but d'augmenter sa durée de vie. Déterminer le temps de validité d'une réponse pour chaque ressource doit tenir compte de multiples paramètres et la littérature n'offre pas de méthodes explicites pour déterminer des temps de validité efficaces pour une configuration donnée. Une modélisation du problème donnée sous la forme d'une optimisation multi-objectifs permettra de fournir un ensemble de solutions admissibles optimales.

Afin de documenter et exécuter de manière reproductible nos expériences nous avons construit un framework d'expérience reproductible appelé Makesense permettant de documenter, d'exécuter et d'analyser l'ensemble d'une expérience effectuée sur un LLN. Effectuer une expérience avec des LLNs met en jeu de nombreux logiciels et des procédures qui sont longues et fastidieuses lorsqu'elles sont réalisées manuellement ce qui rend une expérience difficile à reprendre surtout si elle n'est pas ré-effectuée par la ou les mêmes personnes. Fonctionnant à la fois en simulation et sur nœuds réels, Makesense permet d'obtenir un framework d'expériences reproductibles tout en utilisant des outils classiques et largement utilisés par la communauté scientifique.

Remerciements

Je tiens à remercier Andrezj Duda et Nathalie Mitton pour avoir accepté de rapporter ma thèse. Je veux remercier également tous les membres de jury ainsi que mes directeurs de thèses Claude Chaudet, Jeremie Leguay et Paolo Medagliani pour leur aide et leur encadrement durant cette thèse.

Merci à Marc-Oliver, Fabien et Jawad pour les relectures de mon manuscrit. Merci à Grégoire pour les discussions sur les méta-heuristiques.

Je veux remercier tous les collègues de LINCS : José, François, Ludovic, Alonso, Julian, Ahlem, Nivine, Sameh, Ghida, Riad, Jordan, Ahlem, Nihel, Andrea et Yu-Ting pour l'ambiance chaleureuse.
Merci à l'équipe de TAI : Kevin, Jawad, Mathis, Mathieu, Damien, David, Filippo, Bruno, Farid, Raphaël, Hicham, Julien, François-Xavier, Mario, Nicolas, Marie-Noëlle, Geoffroy, Mathias, Romain, Lionel et Catherine pour m'avoir tant apporté pendant ces trois années.

Merci à Simon Duquennoy et à son équipe pour m'avoir accueilli au SICS.

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 veux remercier ma famille et mes amis qui sont restés à mes côtés pendant ma thèse.

Je voudrais remercier aussi Paris Montagne pour ces années passées à côtoyer 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 ces gens qui m'ont (dé)formé l'esprit.
Enfin je voudrais remercier Milena pour sa patience et sa tendresse qui m'ont grandement aidé à me reconstruire après les épreuves que ces dernières années m'ont apportées et qui m'ont aidé à finir la rédaction de ce manuscrit.

Table des matières

Table des figures

Liste des publications

— Rémy Leone, Paolo Medagliani et Jérémie Leguay. Optimizing QoS in Wireless Sensors Networks using a Caching Platform. *Sensornets 2013*, page 56, Barcelone, Espagne, Février 2013.
— Rémy Leone, Paolo Medagliani et Jérémie Leguay. Optimisation de la qualité de service par l'utilisation de mémoire cache 15èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications
— Rémy Léone, Jérémie Leguay, Paolo Medagliani, Claude Chaudet. Tee : Traffic-based energy estimators for duty-cycled Wireless Sensor Networks. *IEEE International Conference on Communication (ICC)*, page 6749-6754, Londres, 2015.
— Rémy Leone, Jeremie Leguay, Paolo Medagliani, Claude Chaudet, et al. Makesense : Managing reproducible wsns experiments. *Fifth Workshop on Real-World Wireless Sensor Networks*, 2013.
— 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.
— Rémy Léone, Jérémie Leguay, Paolo Medagliani, and Claude Chaudet. Demo Abstract : Automating WSN experiments and simulations. In *EWSN*, 2015.
— Fadwa Boubekeur, Lélia Blin, Remy Leone, and Paolo Medagliani. Bounding degrees on rpl. In *Proceedings of the 11th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, pages 123–130. ACM, 2015.

Journaux

— 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.

Invitations

— Invitation au colloque R4 : Retour d'expéRiences sur la Recherche Reproductible. Le Studium - Centre International Universitaire pour la Recherche - Orléans 3 et 4 Décembre 2015

Chapitre 1

Introduction

We always overestimate the change that will occur in the next two years and underestimate the change that will occur in the next ten.

Bill Gates

Sommaire

Internet s'est développé au cours des dernières décennies, partant d'un petit réseau académique pour devenir ubiquitaire et mondial [154]. La conception de ce réseau avait pour hypothèses de départ les contraintes technologiques de l'époque : des ordinateurs, toujours connectés, alimentés en énergie en permanence et disposant d'assez de capacité pour gérer le trafic réseau reçu. L'augmentation croissante du nombre d'appareils connectés comme des ordinateurs, tablettes ou smartphones est venue s'ajouter autour des réseaux de cœur pour l'étendre et proposer de nombreux usages plus mobiles et orientés vers les utilisateurs finaux apportant services et fonctionnalités [61].

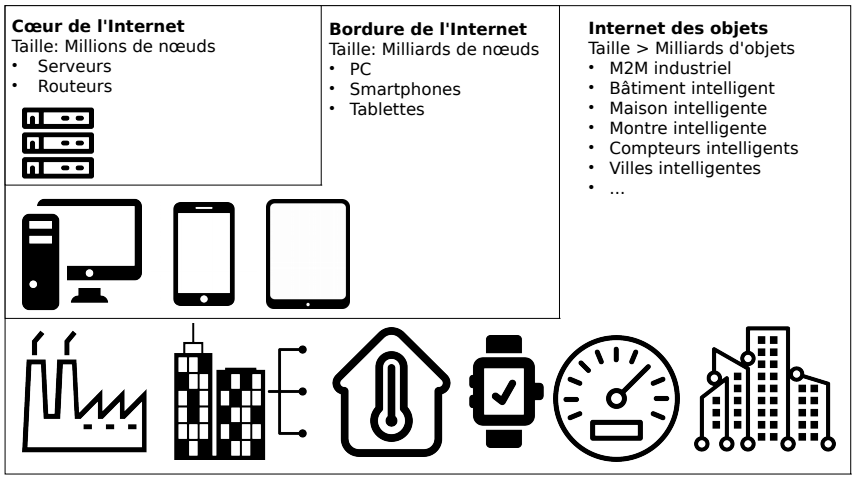


FIGURE 1.1 – Internet des objets (IoT)

L'Internet des objets (appelé Internet of Things (IoT) en anglais) se construit autour des réseaux préexistants montrés sur la Figure 1.1 et vient ajouter à des objets existants des fonctionnalités de connectivité à l'Internet. L'objectif de l'IoT consiste à ajouter une connectivité à Internet pour un coût négligeable à des objets afin de permettre l'émergence de nouveaux services et de simplifier leur utilisation. Cette tendance est rendue possible par la réduction constante des besoins énergétiques, de la taille, du prix et du poids des processeurs et des capteurs qui a permis d'ajouter une connectivité sans fil à un grand nombre d'appareils [99]. Le nombre de machines et leur versatilité sont en pleine croissance [72] de même que les marchés pour ces appareils aussi bien dans le secteur privé que grand public. Les domaines d'applications étant vastes il s'ensuit que les applications proposées seront très diversifiées et toucheront l'ensemble de la société.

Pour fonctionner, l'IoT repose sur des capteurs envoyant des informations sur leur environnement en temps réel vers des services à valeur ajoutée que les utilisateurs finaux consultent. Certains de ces capteurs seront intégrés dans des systèmes embarqués devant fonctionner de manière fiable sur de grandes périodes, mais avec des ressources matérielles faibles et des batteries limitées. Afin d'augmenter leur couverture réseau, ces systèmes sont connectés les uns aux autres pour former des réseaux maillés. À l'interface de ces réseaux maillés et des réseaux usuels se trouve une passerelle dont l'objectif est de masquer l'hétérogénéité de ces nœuds et leurs contraintes spécifiques tout en leur offrant une connectivité native au reste du réseau.

L'objectif de cette thèse est d'étudier les fonctionnalités que l'interface entre un réseau de capteurs et un réseau conventionnel peut offrir afin d'améliorer les performances et la fiabilité de fonctionnement du réseau visé.

Ce chapitre introductif commence par introduire une taxonomie des machines étudiées (1.1) puis rappelle les enjeux et domaines d'application (1.2) et les défis apportés par cette nouvelle tendance (1.3). Les contributions de cette thèse (1.4) et le plan (1.5) seront ensuite présentés.

Taxonomie des machines en IoT

Un réseau de capteurs (aussi désigné sous le terme plus spécifique de Low-Power and Lossy Network (LLN) en anglais) désigne un réseau formé par des capteurs et une passerelle communiquant

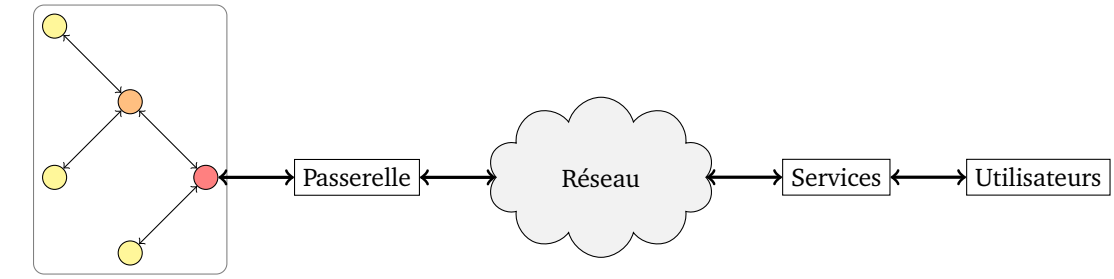


FIGURE 1.2 – Taxonomie des différents nœuds

entre eux sur des connexions radio particulièrement bruitées [72]. Certains de ces nœuds ont un rôle de routeur interne relayant les communications des autres nœuds vers la passerelle (aussi appelé routeur de bordure) afin d'augmenter la couverture d'un réseau sans augmenter les portées de transmission. C'est ce type de réseau qui est considéré dans cette thèse.

Nœuds capteurs

Afin de mesurer l'environnement et d'interagir avec lui, il est nécessaire d'avoir des capteurs et actionneurs simples, représentés à gauche sur la Figure 1.2. Ils sont utilisés dans des contextes variés et répondent à des besoins hétéroclites [194]. Ils transmettent et reçoivent des messages courts comme une mesure d'un capteur ou l'ordre de déclenchement d'un actionneur et doivent fonctionner de manière fiable durant des périodes longues.

Les améliorations technologiques sur les composants sont généralement utilisées pour baisser les coûts de production et d'exploitation, notamment en énergie, plutôt que pour améliorer les performances [135], ainsi leurs capacités restent limitées. Autrement dit, les architectures matérielles sont réduites (processeur, mémoire), la consommation énergétique doit être faible et la batterie doit tenir sur de longues périodes [194]. Pour qu'ils puissent avoir une grande durée de vie avec des batteries simples (piles classiques) les nœuds se mettent en veille autant que possible pour préserver leurs réserves d'énergie.

Afin de masquer l'hétérogénéité des nœuds et des protocoles spécifiques qu'ils utilisent pour se parler, ces nœuds ont besoin de passerelles pour communiquer avec l'extérieur.

Passerelles

Ces passerelles (parfois aussi appelées routeurs de bordure) sont en charge de concilier un écosystème d'appareils hétérogènes, de collecter les données des nœuds capteurs et d'offrir une interface vers eux. La nature hétérogène des capteurs conduit les passerelles à supporter différents types d'interfaces (par exemple : Courant Porteur en Ligne (CPL), Bluetooth, cellulaire ou encore Wi-Fi) et protocoles réseau. Elles masquent ainsi la spécificité des capteurs derrière une interface présentant les ressources avec un formalisme commun.

Les passerelles sont plus performantes que les nœuds capteurs et ont un rôle de médiateur entre différentes technologies, comme montré sur la Figure 1.2 . Du fait de leur position, les passerelles disposent de beaucoup d'informations sur les nœuds capteurs et peuvent en tirer parti entre autres pour établir et optimiser des tables de routage [197] ou découvrir les services offerts par les capteurs [34].

Ces fonctionnalités peuvent s'ajouter à d'autres fonctions (Pare-feux, supervision) et seront détaillées dans le chapitre 2.

Infrastructure de services

Bien que les passerelles assurent la connectivité vers les nœuds, l'intégration de ces données est faite au niveau de serveurs distants [11, 12] pour faciliter le passage à l'échelle de sources d'informations multiples. Ces serveurs peuvent être physiquement localisés dans un même réseau (par exemple dans les scénarios domotique) ou bien chez un fournisseur de services tiers pour en simplifier l'usage (accès depuis l'extérieur, fiabilité renforcée, etc.) Ils sont représentés à droite sur la Figure 1.2 et représentent le dernier chaînon avant l'utilisateur. On y trouve les services à valeur ajoutée en charge de traiter, analyser et visualiser les données dans une interface destinée aux consommateurs et utilisateurs finaux.

Il devient alors possible de construire des services basés sur les données collectées en temps réel comme des bulletins météo ou des prévisions de trafic routier [78, 72]. Dans cette architecture, ces fonctions d'intégration seront au plus proche des utilisateurs finaux afin que seules les alarmes, exceptions ou notifications pertinentes soient envoyées.

Enjeux & motivations

L'objectif des LLNs 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 système filaire classique peut offrir. Déployés à grande échelle, ces capteurs fournissent des mesures régulières et systématiques de l'environnement permettant des prises de décision plus réactives sur un système complexe. Les LLNs sont utilisés dans des contextes très variés [11, 12]. Les sections suivantes présentent plus en détail les deux principaux champs d'application des LLNs : l'industrie et la ville intelligente.

Applications en milieu industriel

Les unités de production industrielle ont besoin d'avoir un contrôle et une supervision en temps réel aussi fine que possible sur leurs chaînes de production [60, 74]. Ces déploiements mettent en jeu des capteurs hétérogènes et doivent rapporter en temps réel les événements de manière fiable tout en s'acclimatant à des conditions variées (température, radiation, produits chimiques). Les canaux de communications utilisés doivent être fiables, bidirectionnels (on doit pouvoir communiquer avec une machine et elle doit pouvoir répondre sans limites), avec des délais courts et des bandes passantes suffisantes pour des messages compacts.

Les systèmes de télémétrie industriels usuels (Supervisory Control and Data Acquisition (SCADA)) proposent une solution à ce problème. Ils fonctionnent le plus souvent en filaire et dans de grands déploiements cette solution peut être difficile ou coûteuse à mettre en place [11]. De plus, les interconnexions entre les différents systèmes de télémétrie, qu'elles soient physiques (câble) ou bien logicielles sont très souvent propriétaires, incompatibles avec les machines de la concurrence et aux fonctionnalités limitées et peu évolutives.

Les LLNs utilisent des protocoles interopérables et des standards ouverts entre tous les équipements de différents constructeurs, ce qui permet de disposer d'une interopérabilité entre des systèmes très hétérogènes [164, 188]. Afin de réduire les coûts d'interopérabilité, on assiste désormais à la croissance des communications Machine to Machine (M2M) reposant sur des standards réseau compatibles avec ceux utilisés sur Internet et proposant une normalisation des interfaces applicatives.

Agriculture et élevage intelligent

Les exploitations agricoles sont en croissance du point de vue de leur superficie et de leurs productions, elles adoptent de plus en plus des méthodes de production venues du monde industriel [161]. Les besoins sont nombreux : surveiller en temps réel l'état des plantations, l'irrigation, la présence de pesticide ou de produits chimiques, l'acidité des sols et des conditions météorologiques [192]. Pour ces besoins, il est nécessaire de disposer de méthodes de relevés fiables de l'environnement en temps réel permettant de prendre des décisions de manière automatisée à grande échelle. Les LLNs offrent toutes ces possibilités et permettent par exemple de surveiller l'état de santé d'un animal, s'assurer que son état et son environnement sont conformes avec les normes en vigueur et s'assurer de sa traçabilité ainsi que de sa localisation, ce qui rend les contrôles sanitaires plus rapides, fiables et systématiques sur de grandes échelles, cela afin d'éviter les intoxications alimentaires [189].

Gestion de bâtiment - Domotique

La domotique et la gestion de bâtiment sont des secteurs en pleine expansion [124, 59]. 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'allumage des pièces, la fermeture des portes ou la détection de cambriolage [122]. Le faible coût et la facilité de déploiement des LLN permettent de les installer rapidement afin d'améliorer le confort et le contrôle à distance d'un grand bâtiment ou d'une habitation. Une fois ce contrôle disponible, il est possible de contrôler le chauffage de manière beaucoup plus automatisée sans l'intervention d'un technicien sur le site et de contrôler plus finement la dépense énergétique pour chauffer un bâtiment, ce qui peut conduire à des économies importantes [58].

Applications pour les villes intelligentes

Les villes deviennent de plus en plus peuplées et la modernisation de leur fonctionnement permet des économies et des gains de qualité de vie pour chacune d'entre elles [29]. La variété des déploiements et des autorités mises en jeu rend ce contexte différent de celui des applications industrielles où une autorité unique prend la décision d'intégrer un nouveau système à l'existant. Le but de la ville intelligente est de permettre le déploiement de systèmes à l'échelle d'une ville ou d'un territoire afin de faciliter la vie de ses habitants [81].

Voirie

Les villes ont par exemple besoin de pouvoir réduire efficacement le temps passé par un automobiliste pour trouver une place de stationnement dans le but de réduire le carburant utilisé et limiter les bouchons de circulation. Les systèmes de Smart parking [128] sont un bon exemple de déploiement urbain permettant d'avoir en temps réel la disponibilité des places de parking. Ces systèmes sont mis en place en combinant des capteurs détectant des places disponibles sur la chaussée et envoyant ces informations vers des serveurs qui les redistribuent vers des utilisateurs finaux en temps réel. La gestion du trafic automobile et des piétons peut également être plus efficace quand le trafic est fluidifié par des feux de circulation qui s'adaptent pour le gérer au mieux [46, 62].

Enfin, ces systèmes ont l'avantage de réduire les coûts de fonctionnement sur des temps courts et peuvent être justifiés facilement par la réduction des dépenses publiques et la qualité de vie améliorée pour les habitants de la ville [172, 107].

Sécurité et Urgences

La surveillance d'infrastructure comme des voies de chemin de fer, des frontières ou bien des convois sont nécessaires afin d'assurer une sûreté de fonctionnement pour les usagers et les biens. Dans ce genre de situation, les systèmes doivent résister à de nombreuses attaques ou altérations de leur environnement tout en étant capables de répondre en temps réel en cas d'attaque et d'être fiables tout au long du déploiement. Des systèmes utilisant des LLNs sont mis en œuvre dans des cas de protection et de surveillance de zone dangereuses ou sécurisées [28]. Elles permettent la rapidité de déploiement sans fil, la mobilité et la détection en temps réel à moindre coût. Ces approches peuvent être couplées avec celle des villes intelligentes dans le cas de surveillances des ponts, des routes ou de grandes structures par leurs vibrations [97] afin de pouvoir altérer le trafic quand une partie de la route n'est pas sure. En outre, la surveillance systématique des infrastructures permet de détecter des problèmes plus rapidement et efficacement. Le déploiement des LLNs permet d'avoir des relevés en temps réel de ces installations. Couplé à des actionneurs, il est possible d'effectuer certaines tâches de maintenance à distance et automatiquement et donc d'éviter de solliciter un technicien.

Environnements intelligents

Détecter un risque environnemental est une nécessité pour protéger efficacement les populations et l'environnement. Afin de surveiller en temps réel une large zone, des moyens coûteux (photos satellites, surveillance aérienne, patrouilles) et peu réactifs sont possibles, mais ne sont pas utilisables par les autorités chargées de cette surveillance pour des raisons de coûts ou de logistique. Les LLNs peuvent permettre un suivi en temps réel sur des zones géographiques larges et à moindre coût afin de permettre aux autorités d'intervenir rapidement quand une alerte est déclenchée. Des déploiements ont déjà été réalisés afin de prévenir des feux de forêt [202], des avalanches et glissements de terrain [181, 6], de permettre la détection des risques sismiques et volcaniques [194] ou bien la supervision de la qualité de l'eau et de l'air dans des terrains industriels ou contaminés [95, 137]. Les scénarios de surveillance de l'eau [199], d'une rivière ou nappe phréatique sont particulièrement pertinents dans le cas de ville ou de zone géographique où l'eau est rare [94].

Défis introduits par les LLNs

Les LLNs sont des réseaux composés de systèmes embarqués caractérisés par une faible consommation énergétique, l'interconnexion native avec Internet via leur passerelle et des ressources limitées sur chaque nœud. Les systèmes embarqués sont devenus, avec la croissance des smartphones, les appareils majoritaires en nombre sur Internet et cette croissance se maintient dans beaucoup de pays [96]. L'objectif voulu par les LLNs est de fournir des mesures en temps réel d'un environnement afin de permettre la création de services autour de ces données mesurées. Cependant, les LLNs disposent de leurs propres défis techniques qui les distinguent des systèmes embarqués usuels comme les smartphones et qui tiennent à leur nature même et à leurs déploiements.

La présentation de ces défis sera découpée en deux axes : l'un portant sur les défis relatifs aux nœuds et l'autre sur ceux liés à la mise en réseau de ces mêmes nœuds.

Défis liés aux nœuds

Les nœuds capteurs sont des systèmes embarqués visant à fournir des mesures sur leur environnement et à exécuter des actions sur celui-ci [100]. Ces nœuds sont composés de capteurs, d'actionneurs et fonctionnent grâce à un microcontrôleur ou bien avec un microprocesseur sur lequel

s'exécute un système d'exploitation spécifique [53]. Ces composants qu'ils soient matériels ou logiciels consomment aussi peu de ressources que possible, visent à être aussi fiables que possible, et cela pour un faible coût de production. Contrairement à d'autres systèmes embarqués comme des smartphones, ils ne cherchent pas à être des systèmes universels ni à devenir plus performants. Les évolutions technologiques visent à réduire leur coût et améliorer leur efficacité énergétique [99]. En plus de ces contraintes spécifiques, d'autres défis viennent s'ajouter lorsque ces capteurs sont déployés dans le contexte spécifique des LLNs.

Hétérogénéité technologique et fonctionnelle

Le grand nombre de constructeurs impliqués dans les LLNs et la multitude des domaines d'applications a conduit à l'émergence de plusieurs standards [17]. Orchestrer un écosystème hétérogène tant en termes de domaines d'applications et de choix technologiques est donc un défi complexe à réaliser.

L'hétérogénéité se situe à plusieurs niveaux : segments (grand public, industriel), chaînes de valeur (opérateur, fournisseur de service), délais de standardisation (time to market), taille du segment et influence géographique. Cela fait que de nombreux standards coexistent et que des outils seront requis pour offrir des interfaces entre ces différents standards. Des organismes nationaux et internationaux se penchent sur les questions d'interopérabilité et de standardisation : Allseen Alliance, Industrial Internet Consortium, ETSI, Open Interconnect, Thread, IPSO Alliance, IEEE...

Cycle de vie

Les fonctions d'un nœud capteur peuvent changer et nécessiter des mises à jour logicielles par exemple pour corriger des failles de sécurité ou ajouter de nouvelles fonctionnalités. Un déploiement rapide des correctifs et des mises à jour est un facteur clé pour éviter l'obsolescence d'un objet connecté et les problèmes de sécurité qui en découlent [141, 190].

Toutefois, il peut être difficile de mettre à jour, par exemple quand le déploiement a lieu dans un terrain difficile d'accès comme un volcan ou une zone de montagne. Des solutions de migrations, de maintenance et de déploiement d'applications à distance sont donc requises [24, 176].

Défis liés au réseau

Les nœuds capteurs disposent d'une faible capacité d'émission radio, ainsi, les mettre en réseau afin qu'ils communiquent entre eux vers une passerelle permet d'améliorer la couverture du réseau tout en maintenant des portées de transmissions faibles. Cependant la mise en réseau des LLNs comporte des défis techniques.

Pertes

Les LLNs utilisent généralement des bandes de fréquences libres (Bande industrielle, scientifique et médicale (ISM)) pour communiquer afin d'éviter les coûts d'une licence pour une fréquence propre. Les canaux étant communs, les liens sont bruités, instables et les pertes de paquets sont donc courantes [15]. De plus, les capteurs émettent avec une faible puissance pour économiser de l'énergie, ainsi, ils sont particulièrement sensibles aux conditions du canal et leur signal peut être écrasé par d'autres signaux émis dans la même bande de fréquence comme du Wi-Fi. En outre des phénomènes d'atténuation dus à la réverbération (Multipath fading) se produisent et perturbent les communications mêmes si elles sont à portée de transmission [153].

En cas de trop fortes pertes et de conditions trop instables sur le réseau, les LLNs peuvent adapter la topologie de routage utilisée pour communiquer plus efficacement avec la passerelle [37]. Ceci

peut provoquer des reconfigurations dans tout le LLN, ainsi, des mécanismes doivent être mis en place dans les protocoles pour garantir la connectivité et la fiabilité des communications entre les capteurs et la passerelle à moindre coût énergétique [3].

Connexions intermittentes

Afin d'économiser autant d'énergie et de bande passante que possible, les nœuds se mettent en sommeil régulièrement. Durant cette période, leur consommation énergétique est significativement inférieure à la moyenne, mais le nœud est alors difficilement joignable [92]. Cela entraîne l'intermittence des liaisons radio entre chaque nœud [52] et implique qu'il est particulièrement difficile de savoir si un nœud est en panne, indisponible ou endormi dans le cas où ces cycles de sommeil ne sont pas déterministes. Les appareils doivent donc gérer eux-mêmes les confirmations et la fiabilité des transmissions, car utiliser un protocole de transport classique apporterait trop de surcoûts en entêtes et retransmissions [163]. Ainsi, des protocoles sans sessions et des architectures robustes aux déconnexions sont privilégiés afin de transmettre des messages dans le LLN.

Besoin de protocoles optimisés

Dans les déploiements classiques des LLNs, les quantités de données échangées sont très réduites et intermittentes afin de préserver l'énergie [180]. Cela requiert d'utiliser des protocoles spécifiques (compact, sans maintien d'état, avec des entêtes compressés) qui réduisent leur impact autant que faire se peut [198, 164, 162].

Les protocoles classiques utilisés dans l'état de l'art pour faire fonctionner l'Internet ont été conçus pour des appareils étant toujours allumés, générant un trafic important et en croissance. En outre, utiliser les protocoles classiques occuperait une quantité importante de la bande passante et engendrerait donc un surcoût non négligeable de consommation d'énergie. L'émergence de ces protocoles justifie l'apparition d'une couche applicative d'interopérabilité afin d'orchestrer les différents nœuds sans se focaliser sur leurs spécificités [184].

Écoute passive en zone dense

De nombreux protocoles d'accès utilisés dans les LLNs sont asynchrones afin d'éviter de mettre en place une signalisation régulière coûteuse en énergie et en bande passante sur chaque nœud. Lorsqu'un nœud reçoit un signal asynchrone, 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 [106].

Ainsi pour des déploiements denses avec un grand nombre de nœuds, il est indispensable d'utiliser des protocoles spécifiques permettant d'avoir aussi bien passage à l'échelle en nombre de nœuds que consommation énergétique minimale lors du décodage des trames émises [3]. Les phénomènes de capture typiques de l'écoute passive sont donc évités et l'énergie n'est utilisée que pour décoder des messages pertinents pour le nœud.

Passage à l'échelle

La croissance du nombre d'appareils connectés à Internet reste forte et l'ajout des nœuds venant des LLNs ne fera qu'accentuer cette croissance, car la baisse continue des coûts de production conduit à des nœuds toujours plus nombreux et moins chers [180, 120]. Cela pose la question du passage à l'échelle de ce type d'installation dans des réseaux qui peuvent déjà être chargés [135, 25, 17]. La question du passage à l'échelle concerne à la fois la coordination des nœuds, le partage de leurs ressources et les besoins d'adressage.

Afin de répondre à un besoin croissant d'adressage, IPv6 a été introduit afin de fournir un espace d'adressage global suffisant. L'émergence des LLNs rend IPv6 encore plus pertinent, car il permet de disposer d'un adressage complet de tous les nœuds sur un réseau commun qu'il soit global ou privé sans utiliser de mécanisme intermédiaire coûteux comme un mécanisme de traduction d'adresse réseau (Network Address Translation (NAT)). Cette identification unique permet de facilement fournir une communication sans intermédiaire et simplifie la gestion du réseau par un formalisme commun. De plus, des mécanismes de compression d'IPv6 sont disponibles afin de bénéficier de l'espace d'adressage tout en limitant la taille des entêtes utilisés.

Aperçu des contributions

La passerelle jour un rôle clé dans la mesure, car elle interface le LLN avec le reste du réseau. Elle sert de médiateur et traite donc la problématique de l'interopérabilité des nœuds et des différentes contraintes des capteurs. Sa position dans le réseau lui permet en outre d'avoir une vue précise du LLN et des informations qui y rentrent et en sortent.

Le but de cette thèse est d'exposer comment des fonctionnalités peuvent être ajoutées à l'interface d'un LLN et du reste du réseau pour améliorer l'utilisation des ressources et la connaissance de l'état du LLN. Les contributions proposées dans cette thèse ont pour objectif d'améliorer les performances et la fiabilité d'un LLN.

Mesure implicite de la consommation énergétique d'un LLN

Pour s'assurer que le fonctionnement d'une application est correct, un administrateur a besoin de connaître l'état des nœuds du LLN dont dépend son application. Préserver l'énergie consommée est un problème clé pour ce type de réseau. Il parait donc naturel de mettre à disposition des administrateurs un moyen peu consommateur d'énergie leur permettant d'estimer l'état de leur réseau, d'obtenir une durée de vie estimée et connaître les nœuds les plus sollicités dans une topologie multi sauts.

Mesurer explicitement ces informations n'est pas toujours possible et même quand c'est le cas, il est coûteux de le demander à chaque nœud dans un réseau de grande taille. Ainsi des approches induisant un minimum de transmissions de données peuvent aider à obtenir une cartographie de la consommation énergétique et de l'utilisation de la radio à moindre coût.

Le chapitre 3 montre comment l'observation du trafic routé passant par la passerelle permet d'inférer l'utilisation de la radio, et, dans une certaine mesure, la consommation énergétique. Les limites de cette approche seront également décrites et une correction des biais sera proposée lorsque la supervision active est possible.

Optimisation des ressources du LLN avec un cache intelligent

La passerelle offre une interface vers le LLN et reçoit les requêtes applicatives qui lui sont destinées. Le traitement d'une requête par le LLN est lent, car les nœuds sont peu performants. De plus chaque requête consomme de l'énergie qui est une ressource limitée. Ainsi la passerelle a pour objectif de solliciter le LLN aussi peu que possible tout en répondant de manière fiable aux requêtes.

Une stratégie usuelle pour accélérer les réponses et économiser de l'énergie consiste à maintenir dans un cache au niveau de la passerelle la réponse à une requête donnée pendant un temps de validité précis. Ainsi si la passerelle dispose d'une réponse encore valide pour une requête donnée, elle l'utilise au lieu de solliciter le LLN. Déterminer le temps de validité d'une réponse pour chaque ressource doit tenir compte de multiples paramètres et la littérature n'offre pas de méthode explicite pour déterminer des temps de validité efficace pour une configuration donnée.

Le chapitre 4 propose de déterminer les temps de validité pour réguler le trafic entrant vers les nœuds du LLN en fonction de la durée de vie des nœuds visée par l'administrateur et fraîcheur des réponses attendues. Les objectifs étant antagonistes, il est nécessaire de procéder à un arbitrage afin de trouver un compromis entre différentes solutions. Une modélisation donnée sous la forme d'une optimisation multi-objectifs permettra de fournir un ensemble de solutions admissibles optimales.

Expériences automatisées et reproductibles pour LLNs

Effectuer une expérience avec des LLNs met en jeu de nombreux logiciels et des procédures qui sont longues et fastidieuses lorsqu'elles sont réalisées manuellement. Cela rend une expérience difficile à reproduire surtout si elle n'est pas ré-effectuée par la ou les mêmes personnes. Lorsque de nombreuses étapes sont nécessaires pour obtenir un résultat, documenter et automatiser l'expérience en question autant que possible devient crucial afin qu'elle puisse être reproduite et validée par la communauté scientifique et de s'assurer de leur véracité efficacement.

Les bancs de test (appelés testbeds en anglais) prévus pour les LLNs fournissent de nombreux outils dédiés au lancement d'expériences et à la collecte de résultats [26, 68]. Cependant il n'existe pas de solution complète pour lancer, récupérer et exploiter les données issues d'expériences et de simulations dans un seul outil cohérent et intégré. En outre, l'un des écueils à éviter lors de la conception de tels outils est de définir une architecture trop spécialisée et donc inutilisable dans d'autres contextes, ce qui limiterait d'emblée son impact.

Le chapitre 5 propose un framework d'organisation d'expérience permettant de documenter, d'exécuter et d'analyser l'ensemble d'une expérience sur LLN. Fonctionnant à la fois en simulation et sur nœuds réels, Makesense permet d'obtenir un framework d'expériences reproductibles. Ce chapitre consacré à Makesense illustrera son fonctionnement au travers d'une expérience typique. Il présente entre outre les étapes clés d'une expérience et montre que les choix technologiques ne requièrent pas une implémentation spécifique, mais utilise au contraire des outils classiques et largement utilisés par la communauté scientifique. Enfin, un mécanisme d'intégration continue automatisera l'expérience et apportera ainsi la preuve de sa reproductibilité.

Collaborations extérieures faites durant la thèse

Un aperçu des collaborations extérieures effectuées au long de cette thèse avec des chercheurs extérieurs à notre équipe de recherche est disponible en annexe A.

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.

Le chapitre 2 introduit plus en détail les LLNs. Il présente les hypothèses de travail et les choix faits dans cette thèse pour interfacer un LLN avec d'autres réseaux et les confronte avec ceux couramment trouvés dans la littérature.

Le chapitre 3 montre comment la mesure des temps d'utilisation de la radio dans un LLN permet de prévoir la consommation énergétique implicitement.

Le chapitre 4 expose comment l'utilisation d'un cache applicatif peut être adapté pour optimiser l'utilisation des ressources d'un LLN en modifiant les temps de validité des réponses des requêtes qu'il reçoit.

Le chapitre 5 présente Makesense le framework d'expérimentation utilisé ultérieurement dans les chapitres 4 et 3 pour documenter, reproduire et partager les expériences effectuées sur les LLNs.

Enfin, le chapitre 6 conclue cette thèse et ouvre sur des prolongements possibles.

L'annexe A présentera les collaborations extérieures faites pendant cette thèse et présentera les publications obtenues.

Chapitre 2

Caractéristiques d'une passerelle pour LLN

> We build too many walls and not enough bridges.
>
> Isaac Newton

Sommaire

Ce chapitre considère les différentes solutions techniques existantes pouvant être adoptées notamment à la passerelle pour déployer un LLN. Les choix techniques effectués dans cette thèse seront justifiés en prenant en compte les fonctionnalités souhaitées.

La section 2.1 couvre les contraintes d'interopérabilité au travers d'une présentation des différentes solutions d'accès sans-fil aux nœuds et détaille comment les connecter à un réseau IP classique en leur fournissant un adressage commun.

La section 2.2 couvre les contraintes liées au routage dans les LLN et motive les choix faits dans cette thèse.

La section 2.3 aborde comment des fonctionnalités d'interfaces applicatives efficaces peuvent être déployées sur la passerelle afin d'améliorer la rapidité des requêtes et faciliter leur intégration avec des services déjà existants.

La section 2.4 présente les fonctionnalités de supervision déployables sur la passerelle et comment elles peuvent être appliquées pour connaître l'état des nœuds.

La section 2.5 conclut ce chapitre, récapitule les choix techniques effectués et introduit les contributions relatives à la passerelle couvertes dans les chapitres suivants.

Technologie sans-fil et adressage

Comme expliqué dans le chapitre 1, les LLNs mettent en jeu des nœuds hétérogènes avec des architectures matérielles très variées. Les spécificités propres à chaque nœud doivent être masquées pour les administrateurs et fournisseurs de services afin qu'ils puissent se concentrer sur le développement de service à valeur ajoutée plutôt que sur les particularités de chaque nœud. Dans cette thèse, on considère donc que cette opération est déportée sur une passerelle qui est plus adaptée en raison de sa position d'interface et des ressources supplémentaires dont elle dispose.

D'un point de vue matériel, une passerelle est un équipement réseau disposant d'au moins une interface connectée au LLN et d'au moins une interface connectée à un réseau classique fonctionnant par exemple avec IP. Cette passerelle dispose de plus de ressources qu'un nœud capteur et peut rester éveillée en permanence. Elle peut être administrée via une interface graphique comme une interface web.

Communication vers LLNs

Des solutions filaires comme le Power over Ethernet (PoE) permettent de fournir alimentation électrique et connectivité aux capteurs. Cependant, ces solutions filaires peuvent être très coûteuses et difficiles à déployer, notamment dans de grandes zones ouvertes et difficiles d'accès.

Les technologies sans-fil apportent une solution efficace et pratique de connectivité pour l'IoT en utilisant pour leurs transmissions des bandes de fréquences ISM qui présentent l'avantage d'être libres de droits et ne nécessitent donc pas de licence. Les LLNs s'appuient sur ces technologies sans-fil afin de simplifier les déploiements et réduire leurs coûts opérationnels.

Plusieurs solutions sont disponibles [169], chacune d'elle offrant un compromis entre la bande passante, la portée et la consommation énergétique des équipements mis en jeu. On peut distinguer deux grandes catégories de technologies sans-fil : d'une part celle offrant un accès de longue portée (plusieurs kilomètres) et d'autre part celle offrant un accès de courte portée (moins d'un kilomètre).

Accès longue portée

Les accès de type cellulaire permettent de fournir un accès Internet à un grand nombre d'appareils mobiles [96]. Cependant, ils ne sont pas adaptés au cas des LLNs, d'une part en raison du prix de fabrication des composants radios dont ils ont besoin, du coût de l'accès (licence, abonnement auprès d'un opérateur) et d'autre part par la consommation énergétique qu'ils induisent. Ainsi des technologies plus adaptées aux LLNs sont requises pour offrir des accès longues portées.

Les technologies dites Low-Power Wide Area Network (LPWAN) permettent de transmettre un volume de trafic faible [1] sur des distances longues [2] à un faible débit [3] [200].

Ces technologies offrent un jeu de compromis clair : peu de débit, une grande portée, une bonne pénétration [4], un coût de composant radio faible [5] et qui consomme très peu d'énergie.

Cependant, ces technologies présentent plusieurs limitations. Les stations de base lorsqu'elles émettent utilisent une puissance supérieure à celle autorisée pour les usages privés (four à micro-ondes, Wi-Fi, etc.) sur les bandes ISM [157]. Dans ces conditions, les autorités régulatrices de l'utilisation des bandes ISM obligent les stations de base à transmettre au plus 1 % de leur temps [186]. Ainsi, les liens descendants sont limités bien que présents car les liens des LPWANs sont bi-directionnels.

De plus, les bandes de fréquences ISM utilisées étant étroites, les liens descendant et montant sont très proches. Ainsi, les stations ne peuvent pas actuellement transmettre vers les objets tout en écoutant avec le même niveau de sensibilité les autres nœuds de leur cellule. Donc, dans le cas où il faut fournir un service de lien descendant (par exemple pour acquitter un paquet, mettre à jour un objet, ou déclencher des actions), l'utilisation de ces technologies doit se faire dans un contexte où le nombre de stations est surdimensionné afin d'offrir une bonne qualité de service. Par conséquent, si le nombre d'objets devient grand, il devient nécessaire de densifier le nombre de stations de base pour fournir un canal de communication satisfaisant des stations vers les objets, ce qui augmente les coûts de déploiement.

La 5G, annoncée pour l'horizon 2020, entend regrouper et intégrer ce type de technologies longue portée pour une utilisation plus efficace des bandes de fréquences qui seront alors disponibles.

En pratique, les LPWANs sont orientés vers des applications où les objets publient des messages sans confirmation de leur réception (par exemple : un relevé de compteur effectué quotidiennement) et reçoivent encore moins de trafic de la part des stations de base. Ces applications principales concernent les relevés d'informations d'équipements et de télémétries journalières.

Accès courte portée

Les technologies sans-fil de courte portée offrent une alternative permettant d'avoir une bande passante et un débit plus important tout en offrant une meilleure fiabilité des transmissions. Cependant l'utilisation de ces technologies impose un nouveau jeu de compromis.

Les nœuds peuvent être loin de la passerelle qui les connecte au reste du réseau, ainsi l'utilisation de topologies de réseau maillé est nécessaire afin d'offrir une connectivité à des nœuds étant loin de la passerelle.

Les bandes ISM sont très chargées par de multiples appareils comme les micro-ondes ou bien d'autres appareils utilisant le Wi-Fi. Ainsi il est nécessaire pour des nœuds d'un LLN de disposer de mécanismes pour éviter d'avoir leurs signaux écrasés par d'autres stations émettant avec des puissances plus fortes.

D'autre part, les signaux peuvent suivre plusieurs chemins entre deux nœuds en raison des réflexions causées par l'environnement. Ainsi un même signal peut arriver avec des amplitudes différentes et des phases différentes pouvant causer des interférences et des pertes (appelé en anglais "multipath-fading"). Ces conditions sont très instables et dépendent de l'utilisation d'une fréquence donnée, dans un environnement donné, à un temps donné.

Les technologies sans-fil de portée courte sont nombreuses, et seules les plus représentatives seront abordées dans les paragraphes suivants.

1. Environ 200 octets (typique) à 5000 octets (maximum) émis par jour avec une charge applicative utile de l'ordre de 12 octets
2. 5 à 40 kilomètres en champ libre
3. Débit de 0.250 à 11 kb/s en fonction de la technologie radio
4. Important pour des objets situés dans des caves comme des compteurs électriques.
5. Moins de 2 dollars

Wi-Fi

Le Wi-Fi facilite l'utilisation d'Internet sur un grand nombre de périphériques mobiles en proposant un accès natif à un réseau IP avec des débits importants [108]. Il peut fournir des topologie maillée en fonctionnant de manière ad-hoc ou bien des topologie en étoile avec un point d'accès en mode infrastructure.

Le mode la plus économe en énergie du Wi-Fi nommée Power Saving Mode (PSM) permet de former un réseau entre un point d'accès et des stations à portée radio consommant peu d'énergie [9, 182]. Cependant ce mode de fonctionnement impose d'utiliser le Wi-Fi en mode infrastructure avec une topologie en étoile ce qui limite la portée du réseau.

D'autre part, Wi-Fi n'est pas la technologie la plus compétitive pour les LLN car une puce Wi-Fi est coûteuse à fabriquer et offre un débit souvent surdimensionné par rapport aux besoins des nœuds [169]. Ainsi, de nouvelles technologies sans-fil ont donc été conçues afin de transporter des messages à des débits plus modestes tout en consommant moins d'énergie et en restant sur les bandes ISM.

Z-wave

Z-wave permet de connecter des objets dans une topologie maillée principalement dans le secteur domotique. Cette technologie permet d'avoir des communications fiables, avec une faible latence et un débit maximal de 100 kb/s.

Son attrait se trouve dans sa large diffusion dans le secteur de la domotique avec une large gamme de produits compatibles entre eux [167].

Cependant Z-wave ne permet pas de passer simplement à l'échelle [6] sur des scénarios avec un grand nombre de nœuds comme un bâtiment entier, une ville ou un centre commercial qui sont des cas d'utilisation importants des LLNs [167].

Bluetooth Low-Energy (BLE)

BLE est une spécification complète de l'ensemble de la pile protocolaire des couches basses jusqu'à l'applicatif visant les Personnal Area Network (PAN) et qui permet de connecter des objets dans une topologie en étoile sur des portées d'une dizaine de mètres.

Typiquement utilisée pour des appareils personnels (montre et bracelet connectés, accessoires de sport), cette spécification dispose de beaucoup d'atouts : un débit important (1 Mb/s) permettant de passer peu de temps à émettre, des entêtes réduits, une grande efficacité spectrale et un schéma de saut de fréquence pour éviter les canaux chargés typiques des bandes ISM.

Cependant les portées de transmissions sont encore trop limitées et le support des réseaux maillés avec cette technologie n'est pas encore standardisé [166].

IEEE 802.15.4

IEEE 802.15.4 permet de connecter des objets disposant de ressources limitées entre eux par une topologie étoilée, maillée ou arborescente sur des portées de 10 à 50 mètres avec des débits de 250kb/s. IEEE 802.15.4 est utilisée par plusieurs normes [7]. Ainsi, ces normes permettent de disposer d'une large littérature sur ses performances et son fonctionnement. En outre, sa consommation énergétique est faible de même que le coût moyen de ses composants radios physiques [187].

6. Limitation à 4 routeurs intermédiaires pour un réseau maillé, pas plus de 232 nœuds par réseau.

7. Zigbee [7, 170], Wireless HART [110], EnOcean [148], IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) [164] et plus récemment Thread [168]

Protocole	Saut de fréquence	Débit	Topologie	Pile complète
LPWAN	Non	0.250 à 11 kb/s	Étoile	Non
Wi-Fi (PSM)	Non	11 Mb/s	Étoile, Maillé	Non
Z-wave	Non	9.6, 40, 100 kb/s	Étoile, Maillé	Oui
BLE	Oui	1 Mb/s	Étoile	Oui
IEEE 802.15.4	Possible	250 kb/s	Étoile, Maillé	Non

TABLE 2.1 – Récapitulatif des protocoles sans-fil

De plus, des techniques basées sur des sauts de fréquences sont intégrées dans les spécifications les plus récentes de IEEE 802.15.4 et permettent d'augmenter sa fiabilité et de mitiger les problèmes dus au multi-path fading [3].

Conclusion intermédiaire

Le tableau 2.1 offre un récapitulatif des technologies sans-fil considérées. Après analyse des différentes solutions possibles, IEEE 802.15.4 est le protocole qui apparaît comme le plus adapté pour construire des réseaux maillés permettant de pallier efficacement la faible portée des nœuds en augmentant la couverture et cela sans limites de nœuds.

Pour cette raison, le travail effectué dans cette thèse s'appuie exclusivement sur IEEE 802.15.4.

Plan d'adressages dans un LLN

D'un point de vue fonctionnel, dans un réseau maillé, on distingue les hôtes simples représentés par un H sur la Figure 2.1 et les routeurs représentés par un R. Les hôtes envoient et reçoivent des messages, mais n'en transfèrent pas pour le compte d'autres nœuds. Les routeurs font suivre des paquets entre leur origine et leur destination. Il est à noter qu'un routeur peut également recevoir et envoyer des messages et assurer les deux fonctions (routeur et hôte).

Au vu de cette répartition des rôles dans un LLN, il y a essentiellement trois types d'architectures possibles représentées sur la Figure 2.1.

La plus simple consiste en un LLN connecté de manière ad hoc. Les nœuds hôtes envoient des informations que les nœuds routeurs relaient vers leurs destinations. Ce type de déploiement est par exemple utilisé dans des terrains sans connectivité avec un autre réseau [194].

Le second type de déploiement utilise une passerelle pour assurer le rôle de routeur de bordure entre le LLN et un réseau local. Ce type de déploiement peut par exemple se retrouver dans un scénario domotique dans lequel tous les équipements communiquent vers la même passerelle.

Enfin un dernier type de déploiement étend le scénario du routeur de bordure pour prendre en charge le cas où des nœuds peuvent avoir plusieurs passerelles vers un réseau commun. Ce type de déploiement peut être pertinent dans le cas où les nœuds capteurs sont mobiles et doivent pouvoir changer de passerelles tout en restant connectés au réseau et en ne changeant pas leur adresse.

Toutes ces architectures requièrent un adressage de chaque nœud afin de pouvoir garantir des communications en unicast qui sont importantes pour économiser de l'énergie et du débit par rapport à des communications broadcast.

IEEE 802.15.4 fournit un identifiant (nommé Extended Unique Identifier (EUI) en anglais) pour chaque nœud du LLN qui est sous la forme d'un EUI-64 dans sa version "longue" et qui est fournie par le constructeur. Un identifiant plus compact EUI-16 peut être construit à partir du précédent pour réduire la place occupée par ces adresses dans les entêtes. Les protocoles basés sur IEEE 802.15.4

utilisent ces identifiants pour construire un plan d'adressage et une passerelle est requise pour faire une éventuelle traduction entre l'adressage du réseau standard et celui utilisé dans un LLN.

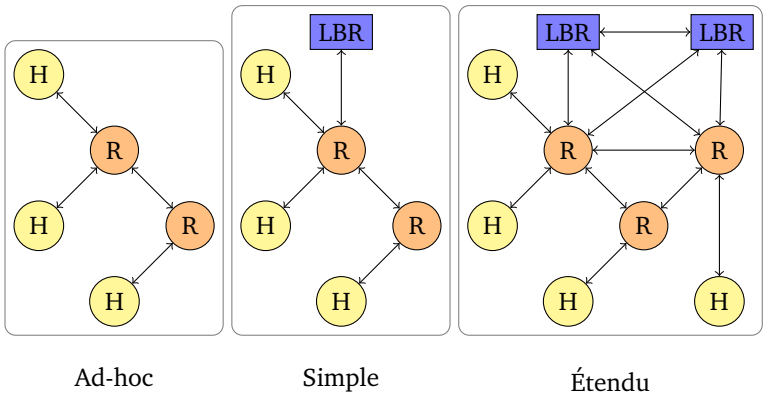


FIGURE 2.1 – Architecture réseau des LLNs

IPv4

IP étant le protocole réseau le plus utilisé aujourd'hui, il est un candidat naturel pour fournir un adressage sur un LLN qui soit interopérable avec un réseau existant. Ainsi une solution consisterait à affecter une adresse IPv4 à chaque nœud.

Toutefois, IPv4 n'est pas pleinement satisfaisant principalement pour deux raisons. D'une part la pénurie d'adresses IPv4 [49] ne permet pas d'assigner une IP publique à chaque nœud ce qui est dommageable dans le cas où des scénarios de connexions directes sont requis. Une solution consisterait à utiliser des NATs [8] afin de pallier le manque d'adresses publiques allouées. Cependant, des NATs requièrent de reconstruire des paquets et ce traitement dégraderait les performances en plus d'être complexe et d'invalider la connexion de bout en bout directe.

D'autre part, les en-têtes IPv4 sont relativement grands, ce qui signifie plus d'informations à transmettre et donc plus d'énergie à consommer pour transmettre un message. Ceci pourrait être clairement amélioré puisqu'un en-tête IPv4 véhicule des informations qui ne sont pas pertinentes pour un LLN (type de service, padding, etc.) [164].

IPv6

Une solution alternative consiste à utiliser IPv6 afin de disposer d'un vaste espace d'adressage rendant l'utilisation d'un NAT superflue et qui permettrait de généraliser les connexions de bout en bout sans intermédiaire. Cependant l'utilisation d'IPv6 comporte des inconvénients dans le cas des LLNs. L'entête de taille fixe d'IPv6 (40 octets) est plus large que celle d'IPv4 (20 octets), d'autre part, IPv6 définit un Maximum transmission unit (MTU) *minimal* de 1280 octets afin d'éviter les problèmes de fragmentation et de recomposition de paquets. Or IEEE 802.15.4 ne peut fournir au maximum qu'une trame de 127 octets, ainsi les paquets IPv6 venant de IEEE 802.15.4 seraient rejetés par un routeur IPv6 implémentant ces recommandations de MTU minimal.

Actuellement, IPv4 est encore largement déployé et donc les adressages IPv4 et IPv6 vont cohabiter plusieurs années [38]. Si un LLN utilise IPv6 pour son adressage et qu'il doit pouvoir recevoir des instructions envoyées depuis un réseau standard fonctionnant en IPv4 alors la passerelle doit

8. Mécanisme permettant de disposer de plusieurs adresses privées derrière une seule adresse publique.

assurer une conversion IPv4/IPv6 qui peut être effectuée par un mécanisme de traduction (teredo, 6to4, isatap, etc.) [44].

Pour transmettre des messages, il est nécessaire d'indiquer un protocole de transport pour les acheminer. Le protocole de trafic le plus compact en terme d'en-têtes considéré dans le cadre des LLNs est User Datagram Protocol (UDP). En effet, comparé à Transport Control Protocol (TCP), il requiert moins d'espace dans l'en-tête du paquet (diminuant ainsi son coût en énergie) et n'utilise pas d'acquittements (limitant ainsi le nombre de messages et donc la consommation d'énergie). Dans le cas où les messages doivent être acquittés, ils peuvent l'être avec un mécanisme ad hoc situé au niveau de la couche applicative.

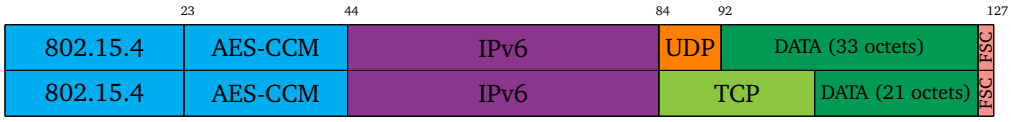


FIGURE 2.2 – Trame 802.15.4 sur IPv6 (sans compression d'en-tête)

Une application directe de cette configuration est représentée sur la Figure 2.2 qui représente une trame IEEE 802.15.4 de donnée ("Data frame" en anglais). En IEEE 802.15.4, la taille maximale de données fournie par la couche physique (Physical Layer Service Data Unit (PLSDU)) est de 127 octets. Utiliser IPv6 sans compression sur cette trame de données de 127 octets impliquerait de laisser seulement 33 octets utilisables pour des messages applicatifs puisque 25 sont pris par IEEE 802.15.4 [9], 40 par IPv6 et 8 par UDP ; soit une efficacité de 26%. TCP offre un rendement encore moins bon, car son entête prend plus de place que celle d'UDP. Afin d'améliorer ce rendement, il est nécessaire de procéder à une compression permettant d'empiler et d'abréger les informations contenues dans l'en-tête IPv6. C'est ce que propose 6LoWPAN.

6LoWPAN

6LoWPAN [103] est une couche d'adaptation d'IPv6 à IEEE 802.15.4 qui permet d'augmenter le rendement des trames.

6LoWPAN définit entre autres, 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 le processus de transfert de trames dans les réseaux maillés [133].

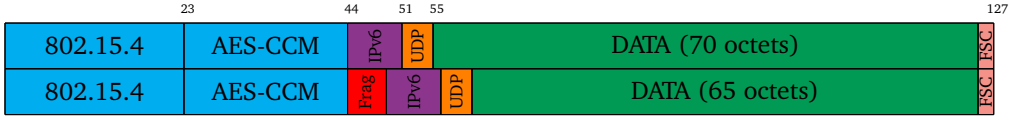


FIGURE 2.3 – Trame 802.15.4 avec entête de fragmentation et Entête IPv6 "Header Compression"

Comme montré sur la Figure 2.3, la compression ne se fait pas au niveau du paquet tout entier car en cas de fragmentation et de perte d'un paquet il serait impossible de reconstituer le paquet original. La compression se passe au niveau de l'entête IPv6 et de l'entête UDP par deux mécanismes de compression [133].

Afin de ne pas utiliser de processus complexes nécessitant de maintenir des états, les compressions sont sans états ou dépendent d'un contexte global à l'ensemble du LLN [164]. 6LoWPAN effectue une compression en retirant entre autres le préfixe IPv6 qui est le même au sein du réseau et en

9. L'entête de sécurité représenté par AES-CCM sur la Figure 2.2 est optionnel et peut consommer jusqu'à 21 octets

compressant les Interface ID (IID) qui sont basées sur l'adresse Media Access Control (MAC) et déjà présentes dans les en-têtes IEEE 802.15.4.

Ainsi, 6LoWPAN permet de disposer de la compatibilité avec les réseaux IPv6 tout en disposant d'en-têtes compressés pour permettre de transmettre des messages applicatifs plus larges pouvant tenir dans la trame entière sans nécessiter de fragmentation.

Routage

Lorsqu'un LLN utilise un réseau maillé pour communiquer (notamment en raison de la faible portée de transmission des nœuds), il est nécessaire de disposer d'un protocole de routage afin que les nœuds routeurs en charge de router les paquets puissent le faire. Cependant les contraintes propres aux LLNs se distinguent de celles des cas classiques de routage disponibles pour les réseaux usuels.

Contraintes du routage dans les LLNs

L'objectif d'un protocole de routage pour LLN est entre autres de minimiser la consommation d'énergie et de supporter les cycles de veille des nœuds en utilisant aussi peu de bande passante que possible.

Faible empreinte sur les ressources d'un nœud

Les routeurs utilisés dans les LLNs sont généralement des nœuds capteurs eux-mêmes afin de simplifier les déploiements. Ainsi, ces nœuds ne disposent généralement que d'une même interface radio pour traiter les paquets qu'ils doivent router pour le compte d'autres nœuds et leurs propres paquets.

De plus, les LLNs mettent en jeu des nœuds fortement contraints ainsi un protocole de routage adapté doit induire aussi peu de signalisation que possible et doit nécessiter un faible espace mémoire.

En outre, le protocole de routage doit supporter les conditions imposées par les économies d'énergies (cycles de veille, bande passante limitée, ...).

Détection des boucles

Chaque équipement maintient dans sa table de routage quel voisin traverser pour joindre une destination. Le protocole de routage doit maintenir ces routes qui doivent refléter la topologie courante du réseau, y compris si celle-ci est modifiée. Dans ce genre de situation, les messages de routage permettent aux différents équipements de corriger leur table de routage. Une telle correction, appelée convergence, peut prendre quelques (milli)secondes. Cependant, rien ne garantit l'ordre dans lequel chaque routeur corrigera sa table de commutation, ni même que toutes les destinations seront joignables pendant que le réseau converge de nouveau. Il arrive parfois que les paquets soient propagés de nouveau de manière incohérente et que des paquets soient propagés le long d'un cycle arbitraire sans jamais atteindre leur destination. On parle alors de boucle de routage.

Dans un réseau classique, une boucle de routage se traduit généralement par une saturation des interfaces mises en jeu dans la boucle (puisque les paquets s'accumulent dans la boucle), ce qui induit de la gigue sur les autres paquets amenés à traverser ces interfaces. Le Time To Live (TTL) permet d'éviter qu'un paquet soit indéfiniment repropagé et donc limite ce phénomène. Toutefois, un paquet dont le TTL expire est jeté et est perdu, ce qui force sa source à le retransmettre (sous réserve qu'elle le détecte).

Dans un réseau LLN une boucle de routage est encore plus négative. D'une part l'utilisation d'UDP au lieu de TCP fait qu'une perte sera généralement définitive. De plus, tout paquet capturé dans une

boucle de routage engendre une perte sèche d'énergie, puisque la plupart du temps le paquet ne sera pas acheminé jusqu'à sa destination.

Pour toutes ces raisons, la détection d'une boucle de routage et son élimination est critique.

Routage optimisé pour les types de trafic typiques des LLNs

Les protocoles de routages classiques comme Open Shortest Path First (OSPF) [42] ou Border Gateway Protocol (BGP) [159] qui sont utilisés dans les réseaux classiques sont agnostiques concernant la direction et l'intensité du trafic émis. Cependant, les LLNs sont des "stub networks", ils disposent de leur réseau, mais ne font pas transiter de paquets pour d'autres réseaux : un nœud ne fera du routage que pour des nœuds appartenant au même réseau. D'autre part, dans un LLN les profils de trafic observés peuvent être répartis entre trois types de trafic.

Multi-point to point (MP2P)

La mission essentielle des nœuds capteurs consiste à envoyer des relevés vers une racine, on parle alors de trafic remontant ou de MP2P. Ainsi la passerelle a un rôle prépondérant pour un protocole de routage puisqu'elle offre un point de sortie pour chaque nœud qui veut envoyer des messages vers l'extérieur. Donc, un protocole de routage idéal pour les LLNs doit être optimisé pour ce type de trafic où les paquets en provenance des nœuds remontent vers une passerelle.

Point to Multi-point (P2MP)

Les nœuds capteurs peuvent aussi recevoir des paquets en unicast de la part de la passerelle on parle alors de trafic descendant ou P2MP. Ce type de trafic est représentatif de l'envoi d'une requête ciblée ou bien du déploiement d'une mise à jour sur un nœud capteur donné.

Point to Point (P2P)

Les cas de trafic P2P doivent être supportés, car ils sont possibles par exemple pour faire une découverte de ressources locales ou effectuer une action, mais ces cas sont minoritaires et le protocole de routage idéal doit concentrer ses optimisations sur les deux premiers types de trafic.

Taxonomie des protocoles de routage

Le routage dans les LLNs a été étudié en profondeur pour le groupe de travail Routing Over Low power and Lossy Networks (ROLL) de l'Internet Engineering Task Force (IETF) [112]. Leurs travaux ont permis d'aboutir à la conclusion qu'aucun protocole de routage disponible alors (2009) ne convenait sans modifications aux spécificités des LLNs et qu'il était donc nécessaire d'en construire un.

Une brève description des principaux types de routage et de leur compatibilité avec les LLNs est fournie dans les sections suivantes afin de comprendre les décisions du groupe de travail ROLL.

Construction des tables

La construction des tables dans un protocole de routage indique comment les routes vont être calculées et les informations qui seront disponibles sur chaque routeur intermédiaire. Il existe deux principales familles de protocoles de routage : le routage à état de lien et le routage à vecteur de distance.

Routage à état de lien

Dans cette approche, chaque nœud a la connaissance complète du réseau qui est un graphe. Afin d'avoir cette connaissance, chaque nœud envoie à tout le réseau les informations à propos de ces liens et des destinations adjacentes. Après la réception d'assez d'états de liens depuis un nombre suffisant de nœuds, chaque nœud calcule un arbre de plus court chemin depuis lui-même vers chaque destination en utilisant l'algorithme de Dijkstra [147].

Ce type de protocole induit une grande quantité de trafic de signalisation notamment quand la topologie des nœuds change souvent. De plus, les nœuds ont besoin de mémoire pour stocker l'état de chaque nœud ainsi ils ne sont pas adaptés pour les LLNs [112].

Il est à noter que ce type de protocole peut éventuellement être utilisé sur les routeurs de bordure qui ont suffisamment de capacité en bande passante et en mémoire pour communiquer aux éventuels autres routeurs de bordure sur le réseau, les routes disponibles vers le LLN.

Routage à vecteur de distance

Les protocoles de routage à vecteur de distance sont basés sur l'algorithme de Bellman-Ford [147]. Dans cette approche, chaque lien a un coût via une métrique préalablement définie. Quand un paquet est envoyé entre deux points, le chemin de plus faible coût est choisi. La table de routage de chaque routeur garde en mémoire les routes des destinations qu'elle connaît avec le coût de chemin associé.

Les informations de routage sont mises à jour soit de manière proactive soit réactive en fonction du protocole de routage choisi. En raison de sa simplicité, de sa faible signalisation et de son adaptabilité, le routage à vecteur de distance est généralement appliqué aux LLNs [145, 83, 197].

Réactivité

La réactivité d'un protocole de routage désigne comment les routes sont entretenues et découvertes au cours du temps. Il existe deux modalités de réactivité pour le routage : le routage réactif et le routage proactif.

Routage réactif

Les protocoles de routages réactifs stockent peu ou pas d'information de routage, ils découvrent dynamiquement les routes lorsqu'elles sont demandées. Un processus de découverte de route est alors exécuté lorsqu'un routeur reçoit un paquet à router avec une destination qu'il ne connaît pas. Ces protocoles sont notamment utilisés dans des réseaux très dynamiques, parmi ces protocoles on peut citer Ad-hoc On-demand Distance Vector (AODV) [145], avec dans le cas des LLN le protocole de routage utilisé par Zigbee qui est dérivé d'AODV ou plus récemment Load-ng [2].

L'avantage de cette approche est que la signalisation de routage n'apparaît que quand elle est demandée, ce qui est utile dans des cas de réseaux ad hoc où la topologie change fréquemment et où les communications sont essentiellement P2P. Cependant, cette approche est coûteuse dans les cas où la topologie change peu. De plus, il est souhaitable a des fins de supervision d'avoir une vue aussi récente que possible sur la topologie du réseau.

Routage proactif

Un protocole de routage proactif construit les routes nécessaires sur chaque nœud avant que les routes soient demandées. Ainsi ces protocoles préparent proactivement toutes les routes possibles ou probables vers une destination. La plupart des protocoles utilisés dans le routage inter-domaine ou intra-domaine utilisent un protocole proactif quand la topologie est stable. D'autres exemples

peuvent être trouvés par exemple dans des cas plus dynamiques comme Optimized Link State Routing protocol (OLSR) [36].

L'avantage de cette approche est que les routes sont immédiatement disponibles quand elles sont nécessaires, cependant cette disponibilité se paye par un coût de signalisation important notamment quand la topologie change fréquemment.

Routing Protocol Layer

ROLL a construit Routing Protocol Layer (RPL) [197] afin de fournir un protocole de routage disposant des fonctionnalités nécessaires et adaptées aux contraintes spécifiques des LLNs spécifiées dans leur cahier des charges [112].

RPL est un protocole de routage proactif à vecteur de distance dédié aux LLNs. Il construit et maintient une topologie de réseau sous forme d'un graphe acyclique dirigé vers une destination (Destination-Oriented DAG (DODAG) en anglais) ayant comme racine une ou plusieurs routeur de bordure (LoWPAN Border Router (LBR) en anglais). Les paquets transmis par les nœuds sont transmis par les liens du Directed Acyclic Graph (DAG). RPL permet le trafic MP2P (trafic montant), P2MP (trafic descendant) et le trafic P2P (trafic entre nœuds).

Ce protocole de routage est standardisé, largement diffusé et est aujourd'hui considéré comme le protocole de routage standard des réseaux LLNs. Ainsi il est le protocole de routage de référence utilisé dans cette thèse.

Structures et signalisation de routage

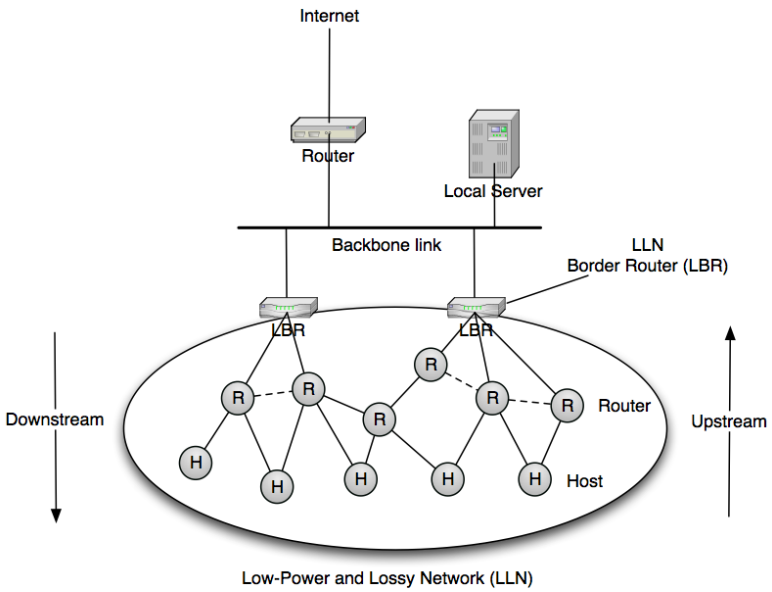


FIGURE 2.4 – Schéma de l'architecture RPL

RPL construit un graphe de routage entre les nœuds du LLN et le routeur de bordure comme illustré sur la Figure 2.4. Le routage effectué par RPL n'est valide qu'au sein du LLN et s'arrête donc aux LBRs. Cette topologie est maintenue en utilisant aussi peu de messages de signalisation que possible. Une fois construite, le protocole de routage maintient des routes montantes (MP2P ou "upstream"

en anglais) et descendantes (P2MP ou “downstream” en anglais) et procède au routage des paquets IPv6. La passerelle peut assurer cette fonctionnalité de routeur de bordure en raison de ses ressources physiques et de sa position de racine.

Les nœuds routeurs disposant d'interfaces vers un autre lien IP sont appelés routeurs de bordure (LBR), il peut y avoir plusieurs LBR pour connecter un LLN au reste d'un réseau afin d'assurer une redondance de liens de sorties souhaitable en cas de pannes. Le rôle de LBR est généralement accompli par la passerelle en raison des multiples interfaces de sorties dont elle dispose.

Les nœuds à l'intérieur du LLN sont soit des routeurs soit des hôtes. Les hôtes ne participent pas au routage et choisissent simplement un routeur par défaut, c'est par exemple le cas des nœuds fortement contraints qui ne peuvent assurer des tâches de routage pour le compte d'autres nœuds. Les routeurs sont eux chargés de router les paquets en provenance d'hôtes ou d'autres routeurs vers leur destination finale.

Construction du routage

Pour le routage montant (MP2P), RPL construit un DODAG qui démarre à partir du LBR. Un rang est assigné à chaque nœud (routeur ou hôte) de telle sorte que le rang augmente à mesure que l'on s'éloigne du LBR. Router un paquet vers le LBR consiste à choisir un voisin avec un rang aussi faible que possible.

Le routage descendant (P2MP) peut être implémenté de manière stockée (“storing-mode” en anglais) ou non stockée (“non storing-mode” en anglais) à l'échelle du DODAG entier. Dans le cas où le routage descendant est non stocké, un routage à la source [85] est fait quand un paquet doit être routé vers un nœud spécifique dans le LLN. Le routeur de bordure indique la séquence de nœuds qu'il faut traverser pour que le paquet arrive à sa destination. Le routeur de bordure apprend ce routage en collectant les Destination Advertisement Object (DAO)s envoyés par chaque routeur vers le LBR et qui indique parents et enfants du routeur en question.

Lorsque le routage descendant est stocké, des tables de routages intermédiaires sont utilisées pour qu'une route partielle soit construite à chaque nœud intermédiaire qui est capable de router le paquet dans son sous-réseau, jusqu'à ce que la destination finale soit atteinte.

Maintenance du routage

Les nœuds exécutant RPL échangent des informations de signalisation pour mettre à jour le DODAG et ainsi mettre à jour de manière proactive la topologie de routage. Il existe plusieurs types de messages de signalisation, celui utilisé pour construire le DODAG est un DODAG Information Object (DIO). Tous les nœuds dans le réseau envoient régulièrement des DIO afin d'indiquer leur rang à leurs voisins.

Dans un LLN, le trafic de routage et de données est limité, ainsi il est nécessaire de détecter aussi rapidement que possible les inconsistances de routage pour les corriger et éviter d'attendre un paquet de signalisation pour lancer une procédure de réparation. Pour cela, un routeur RPL insert une option (nommée RPL Option [84]) dans l'entête de chaque paquet de donnée afin de détecter les éventuelles inconsistances et lancer des réparations ciblées ou globales le cas échéant. Par exemple, 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 inconsistante est détectée et une réparation est déclenchée.

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 tampons.

Limitation de trafic de routage (Trickle)

RPL utilise un mécanisme de timer adaptatif appelé “Trickle” (goutte à goutte) afin de contrôler le débit d'émission des messages de signalisation (DIO) de manière proactive. Trickle double l'intervalle séparant deux émissions successives de messages de contrôle à chaque fois que le réseau est cohérent, et ce jusqu'à une valeur maximale. Ainsi le trafic RPL diminue dans le réseau lorsqu'il est stable et que la topologie change peu. Quand une incohérence est détectée, le timer est réinitialisé à sa valeur minimale et les mécanismes de réparation sont lancés. 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 dans des nœuds disposant de ressources limitées.

Interface applicative

Un protocole applicatif a pour objectif de fournir un accès aux données et services applicatifs proposés. Parmi les exemples les plus connus d'interface applicative, on peut citer l'accès à une page web, le transfert d'un fichier ou encore l'envoi de messages électroniques [154]. Cependant dans le cas des LLNs, l'utilisation principale de cette interface applicative se borne à l'envoi d'instructions courtes et le relevé d'informations sur l'environnement immédiat des nœuds. Ainsi il est nécessaire d'utiliser des protocoles aussi optimisés que possible pour ces usages.

Contraintes de l'interface applicative d'un LLN

Les LLNs doivent s'intégrer aussi facilement que possible aux services d'ores et déjà disponibles afin de faciliter leur adoption. Une des clés de cette intégration et d'avoir une passerelle pouvant offrir une interface commune à de nombreux nœuds. Enfin, il est impératif que cette intégration se fasse en consommant aussi peu de ressources que possible afin de garantir la durée de vie des nœuds et le bon fonctionnement du LLN.

Interface efficace avec les services web

Les services web sont aujourd'hui la façon la plus utilisée pour interroger des serveurs de services [8]. Les ressources y sont désignées par une Uniform Resource Identifier (URI) qui assure l'adressage. L'interaction avec cette URI utilise d'une part le protocole HyperText Transfer Protocol (HTTP), et d'autre part une architecture REpresentational State Transfer (REST) (Architecture client-serveur, absence d'états, etc.) [160].

Afin de s'intégrer aussi facilement que possible à cette architecture, il est souhaitable pour les nœuds capteurs de disposer d'une sémantique proche et de disposer d'URI pour désigner les ressources dont ils disposent. Ainsi lorsqu'un développeur veut utiliser une source d'information en provenance d'un capteur, celle-ci doit être aussi simple à utiliser et proche de ses habitudes que possible.

Disponibilités sur plusieurs plateformes

L'objectif de la passerelle est de masquer les spécificités de chaque nœud capteur derrière un formalisme commun afin de simplifier la tâche des développeurs. Ainsi, une couche applicative idéale doit pouvoir être implémentée sur un large ensemble de plateformes logicielles et matérielles.

Contraintes en ressources

Les nœuds capteurs offrant des services applicatifs disposent de peu de mémoire et de ressources de calculs. Ainsi une couche applicative efficace doit avoir des entêtes aussi limités que possible, doit imposer aussi peu d'utilisation de mémoire que possible (par exemple pour conserver un état) et doit permettre de supporter des notifications afin d'éviter de payer le coût de requête typique des protocoles synchrones et très coûteux pour les LLNs.

État de l'art sur les protocoles applicatifs

Il existe de nombreux protocoles applicatifs pouvant être appliqués aux LLNs [91]. L'objectif de ces couches applicatives consiste à faire un compromis entre jeu de fonctionnalités, interopérabilité et consommation énergétique.

Couches spécifiques à une architecture matérielle

Les piles protocolaires totalement intégrées dans un seul standard (Zigbee [170] ou BLE [166]) proposent leur propre couche applicative standardisée qui peut être prise en charge par la passerelle.

L'avantage de cette approche est qu'elle permet de disposer d'une couche applicative certifiée et d'un fonctionnement interopérable avec tous les appareils supportant la même norme. Cependant ces couches protocolaires ne sont disponibles que sur leur matériel propre et sont donc peu interopérables avec des nœuds hétérogènes.

Message Queuing Telemetry Transport (MQTT)

MQTT [86] est un protocole standardisé par IBM reposant sur un mécanisme de publication-souscription ("publish-subscribe" en anglais, ou "pub-sub" en abrégé) utilisé pour permettre à des abonnés (par exemple des services à valeur ajoutée) de s'abonner aux mises à jour envoyées par des nœuds de manière asynchrone.

L'avantage de ce protocole est d'avoir de nombreuses implémentations, un mode de fonctionnement particulièrement adapté aux notifications et un large support via de nombreuses bibliothèques logicielles [174].

Cependant, ce protocole repose sur TCP qui n'est pas véritablement adapté au contexte des LLNs car les retransmissions qu'il induit consomment bande passante et énergie. De plus, ce protocole n'offre pas d'interface directe vers les mécanismes de services web et leur paradigme REST sans l'utilisation d'un service tiers avec maintien d'un état [41].

Application directe de HTTP

Utiliser HTTP directement sur les nœuds capteurs permettrait d'avoir une pile applicative classique et directement compatible avec les services web usuels. HTTP est largement diffusé, disponible dans de nombreux langages nativement et dispose d'une sémantique puissante via l'utilisation de REST.

Cependant, appliquer cette approche sur les LLNs serait coûteux car l'utilisation de HTTP implique celle de TCP qui n'est pas adapté pour des nœuds très contraints en raison des retransmissions. De plus, cette approche ne supporte pas pour le moment les notifications de type "push" car les services web fonctionnent de manière synchrone or des communications asynchrones sont très courantes dans le cas des LLNs par exemple pour envoyer une notification quand une condition sur l'environnement se produit.

Ainsi il est nécessaire de chercher une synthèse proposant à la fois une intégration efficace avec des services web pour faciliter l'adoption des LLNs comme source de données pour des services web

tout en utilisant un protocole applicatif peu consommateur d'énergie et un support des notifications asynchrones.

Présentation de CoAP

Le groupe de travail Constrained RESTful environment (CoRE) de l'IETF a défini le protocole CoAP [163] pour fournir une interface applicative générique tout en ajoutant des fonctionnalités spécifiques aux contraintes des LLNs comme des entêtes courts et faciles à disséquer, le support de notifications asynchrones, etc.

CoAP est le plus souvent implémenté sur un protocole de transport orienté datagramme tel que UDP afin d'avoir des entêtes courts qui nécessitent peu de bande passante. Ainsi il implémente un mécanisme de transport unicast pour assurer une fiabilité sans utiliser un protocole dédié comme TCP qui peut être plus complexe à implémenter pour les nœuds capteurs et consomme plus de ressources.

CoAP utilise une architecture REST reposant sur des verbes usuels comme `GET`, ou `PUT` permettant ainsi de disposer d'une sémantique riche directement dans le protocole sans nécessiter le maintien d'état en mémoire pour le capteur qui serait coûteux dans le cas de mémoire contrainte.

Support de notifications asynchrones

CoAP supporte les notifications asynchrones afin de permettre à des nœuds capteurs d'envoyer des notifications sur un modèle "push" à un observateur.

En pratique, il n'y a le plus souvent qu'un seul observateur afin de limiter l'utilisation des ressources d'un nœud capteur qui n'a besoin de garder qu'une seule adresse en mémoire. Dans cette architecture, c'est le plus souvent un service qui va recueillir les notifications en provenance des nœuds capteurs et les distribuer vers les abonnés pertinents sans solliciter les nœuds capteurs. Ce mode de fonctionnement peut être accompli par l'utilisation d'un proxy inversé (2.3.4.2).

Traduction HTTP/CoAP

L'un des objectifs de CoAP est de permettre une traduction facile entre une URI utilisée par un service web et celle qui est utilisée pour désigner la ressource d'un nœud capteur facilitant ainsi le travail des développeurs et l'intégration des LLNs.

CoAP n'est pas une compression naïve de HTTP qui serait un standard complexe à implémenter et surdimensionné pour les nœuds capteurs. La conversion entre HTTP et CoAP se fait sur les verbes employés (`GET`, `PUT`, etc.) et les URI (paramètres et options compris). La conversion est sans état et peut être faite efficacement par l'utilisation de proxys traducteurs situés entre les sources des requêtes HTTP et les nœuds capteurs.

De plus, CoAP supporte le traitement complet d'une URI afin de profiter des options qu'elle contient pour répondre plus précisément à une requête (par exemple recevoir une notification si la température dépasse un seuil.)

Traduction, proxy inversé et cache

La passerelle peut offrir une médiation applicative entre les LLNs et les fournisseurs de services à valeur ajoutée. Cette médiation peut être approfondie par la passerelle afin d'offrir des fonctionnalités supplémentaires ayant pour objectif d'améliorer la rapidité du traitement d'une requête et limiter la consommation énergétique des nœuds du LLN.

Traduction

Les nœuds d'un LLN utilisent des protocoles applicatifs hétérogènes et les fournisseurs de services veulent éviter d'implémenter un support pour un grand nombre de protocoles qui serait trop coûteux. La passerelle connaît les spécificités de chaque nœud et peut offrir des traductions de protocoles depuis des protocoles usuellement utilisés dans les services web tels qu'HTTP vers les protocoles utilisés par les nœuds capteurs du LLN. Ainsi les tâches de traduction se font au plus près des nœuds capteurs et permettent de simplifier le traitement des données dans des architectures orientées service.

Proxy inversé

La fonctionnalité de traduction n'est pas suffisante pour fournir un service efficace et sécurisé. En effet, les nœuds capteurs étant contraints, ils ne peuvent recevoir qu'une quantité limitée de trafic. De plus, s'ils sont accessibles depuis un réseau public, il est nécessaire de contrôler finement le trafic reçu afin d'éviter par exemple des attaques par déni de service ou de sommeil [195, 156].

Un proxy inversé ("reverse proxy" en anglais et désigné par Reverse Proxy Cache (RPC) dans le reste de cette thèse) [158] est un serveur qui se pose entre des requêtes provenant de l'extérieur et des services à protéger comme représenté sur la Figure 2.5. Le proxy inversé intercepte l'ensemble des requêtes à destination des nœuds et peut les modifier, les faire passer ou renvoyer des codes d'erreurs.

Le proxy inversé se pose en intermédiaire permettant un gain en termes de fonctionnalité appréciable, car un administrateur peut configurer le proxy inversé pour gérer le trafic applicatif reçu par les nœuds au lieu de le configurer individuellement sur chaque nœud. Ainsi l'administrateur du LLN peut limiter arbitrairement les requêtes transmises aux nœuds pour limiter l'énergie qu'ils consommeraient pour y répondre.

De plus, dans le cas de notifications asynchrones, avoir un proxy inversé qui centralise les notifications est pertinent, car les nœuds capteurs ne doivent envoyer de notifications que vers une seule destination. Lorsque la notification est arrivée sur le proxy inversé il se charge de distribuer cette notification vers les abonnés pertinents. Ainsi la gestion d'un grand nombre d'abonnés est déportée des nœuds capteurs vers le proxy inversé et une économie d'énergie est ainsi réalisée.

Cache

Un cache stocke du contenu afin de l'avoir à disposition sans le redemander au reste du réseau. Une économie de bande passante et de délai est donc réalisée à chaque fois qu'une requête est traitée par l'utilisation d'un cache au lieu d'être envoyée aux nœuds du LLN [195]. À ce titre, le rapport entre le nombre de requêtes traitées par le cache sur le nombre de requêtes total ("hit rate" en anglais) est une métrique importante pour mesurer son efficacité. Il est nécessaire d'avoir une politique de mise en cache, car la quantité de mémoire du cache est limitée, de plus, une information peut devenir obsolète si elle reste en cache trop longtemps.

L'utilisation d'un cache est étudiée dans les LLNs [40] et notamment avec CoAP car elle permet de faire des économies de bande passante et d'accélérer le traitement d'une requête [79].

Conclusion intermédiaire

Comme vu dans cette section, CoAP permet d'offrir une sémantique et un jeu de fonctionnalités riches comme le support de messages asynchrones et la traduction sans état avec HTTP tout en consommant aussi peu d'énergie que possible.

Cette intégration des notifications asynchrones est d'autant plus pertinente que les versions futures de HTTP2 en cours de déploiement supporteront les notifications asynchrones ("push notifications"

en anglais). Ainsi, des traductions sémantiquement complètes pourront être mises en œuvre entre CoAP et HTTP2. Ces aspects sont activement étudiés aujourd'hui et accentuent la nécessité d'avoir des traductions aussi efficaces que possible autour du paradigme REST entre les nœuds et les fournisseurs de services à valeur ajoutée, ce que CoAP fournit.

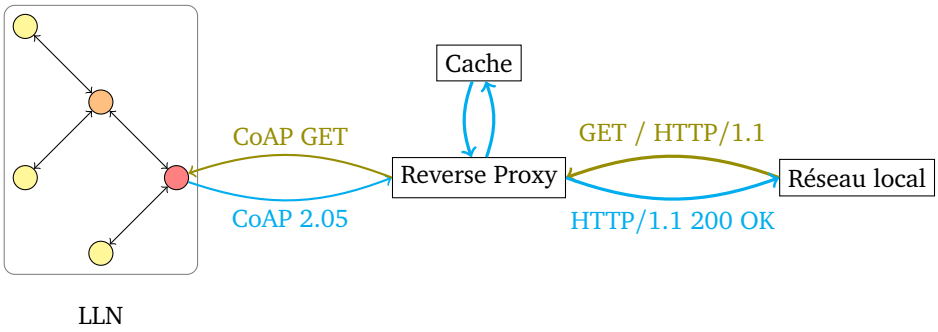


FIGURE 2.5 – Architecture d'un Reverse Proxy Cache (RPC)

CoAP permet de plus l'utilisation de fonctionnalités au niveau de la passerelle afin d'accélérer le traitement des requêtes et économiser de la bande passante. Ces fonctionnalités de traductions entre HTTP et CoAP, de proxy inversé et de cache sont dénommées Reverse Proxy Cache et sont résumées sur la Figure 2.5. Cette figure illustre l'exemple de l'envoi d'une requête HTTP qui est traduite en CoAP avant d'être envoyée dans le LLN. Une fois la réponse CoAP obtenue, elle est retraduite en réponse HTTP puis mise en cache afin d'être réutilisée ultérieurement enfin la réponse HTTP est fournie au client.

En raison de sa pertinence, CoAP sera le protocole applicatif utilisé dans le reste de cette thèse.

Supervision

La supervision désigne la surveillance continue d'un système afin de s'assurer que son fonctionnement et ses performances sont ceux attendus [115].

Cette surveillance est nécessaire pour s'assurer par exemple qu'un système d'acquisition est toujours valide ou bien que ses réserves d'énergies sont suffisantes pour tenir pendant une durée donnée. Un système de supervision peut agir en envoyant une alerte à un humain ou un autre système pour l'informer qu'un LLN rencontre des problèmes et qu'une intervention est peut-être à prévoir. Son objectif est alors de permettre à un administrateur d'identifier aussi rapidement que possible quel est le problème et ses causes.

Dans le cas des LLNs, les métriques recueillies peuvent être relatives à un nœud donné par sa consommation énergétique, l'état du médium, le nombre de retransmissions, la charge moyenne, etc ou bien relatives à l'ensemble du LLNs avec les nœuds les plus sollicités, les nœuds ayant des ressources énergétiques en dessous d'un certain seuil, la latence globale ou les erreurs applicatives, etc.

Toutes ces informations permettent de prendre des décisions motivées et justifiées sur la maintenance du réseau afin de garantir son fonctionnement.

Fonctionnalités recherchées

La supervision couvre plusieurs aspects fonctionnels et administratifs du fonctionnement d'un système. Sa mise en place repose sur plusieurs objectifs fonctionnels et techniques.

Suivi des équipements

Un premier objectif de la supervision dans un LLN consiste à connaître l'ensemble des équipements déployés, leurs fonctionnalités, depuis quand ils sont en fonctionnement. Cette fonctionnalité d'inventaire peut être complexe à mettre en place dans le cas où les équipements changent dynamiquement et sont fortement hétérogènes.

Ce suivi des équipements est utile pour diagnostiquer des problèmes temporels pouvant survenir lors de l'ajout de matériel. Par exemple, l'ajout d'un trop grand nombre de nœuds physiques avec une certaine configuration peut causer une perte de performance selon certaines métriques, etc.

Suivi de la disponibilité

La supervision permet d'obtenir une distinction entre un suivi dans le temps de la disponibilité matérielle et fonctionnelle d'un système qui est nécessaire pour avoir un suivi de supervision efficace. Dans le cas d'un LLN appliqué à un scénario de bâtiment intelligent, une supervision de disponibilité matérielle permet de savoir, par exemple, si un nœud physique donné est disponible. Une supervision de disponibilité fonctionnelle, quant à elle, permet de savoir s'il est possible de mesurer la température sur un étage donné d'un bâtiment.

Cette distinction est essentielle afin d'avoir des alertes de supervision pertinentes et de ne solliciter des interventions humaines coûteuses uniquement pour les problèmes critiques.

Prévision

Un autre volet important de la supervision consiste à analyser les tendances de fond d'un système afin de prévoir son fonctionnement. Prévoir l'évolution d'une métrique permet d'éviter des accidents d'exploitations ou d'autres pannes pouvant survenir lors de l'évolution d'un LLN. En outre, cette prévision permet de séparer les problèmes et les alertes en fonction de leur urgence et de permettre des interventions humaines plus flexibles dans le temps.

Construction de tableau de bord

Les tableaux de bord (en anglais "dashboards") permettent d'agréger et de filtrer les métriques nombreuses et hétérogènes d'un système afin de ne présenter à un administrateur humain que les métriques les plus pertinentes. Ainsi l'administrateur dispose d'une interface qui peut être une visualisation graphique comme une "WeatherMap" [75] capable de représenter visuellement l'état de tous les équipements déployés afin de prendre rapidement une décision à propos d'une situation ou bien demander l'intervention d'un technicien.

Problématiques de la supervision pour les LLNs

Les remontées périodiques d'informations permettent d'obtenir un état sur chaque nœud physique afin de pouvoir l'interpréter. Cependant, ce coût de supervision impacte les réserves d'énergie des nœuds et devrait être réduit autant que possible tout en permettant un niveau minimum de fiabilité sur le déploiement. Ainsi les problématiques de déploiements des LLNs doivent réaliser un compromis entre consommation des ressources et informations gagnées.

De plus, la supervision dans les LLNs rencontre d'autres difficultés techniques pour sa mise en place.

Contraintes physiques et logicielles

Les techniques de supervision usuelles reposent le plus souvent sur l'hypothèse qu'un nœud peut toujours répondre aux requêtes de supervision [191]. Cette hypothèse n'est en général pas validée dans le cas des LLNs.

D'une part, les nœuds d'un LLN ont des ressources limitées : répondre à des requêtes de supervision trop fréquentes peut causer un coût important aussi bien sur leur énergie que sur la bande passante disponible et ainsi compromettre leurs objectifs.

D'autre part, il peut être coûteux d'implémenter une solution de supervision pour l'ensemble des nœuds, car les ressources en mémoire et en architecture matérielle ne le permettent peut-être pas (pas de support pour les nombres flottants, pas assez d'espace mémoire, etc.). De plus, dans le cas où l'administrateur n'a pas la main sur le code s'exécutant sur un nœud, déployer un agent spécifique devient alors inenvisageable.

Grande hétérogénéité

Un autre problème pour la supervision des LLNs provient de la grande hétérogénéité des nœuds physiques et de l'interprétation des métriques. Une métrique au-delà d'un certain seuil peut être acceptable pour certains nœuds, mais pas pour d'autres (par exemple le niveau de batterie, l'espace mémoire flash disponible). Ainsi la définition des alertes et des seuils est complexe et très dépendante d'un contexte donné.

Conclusion

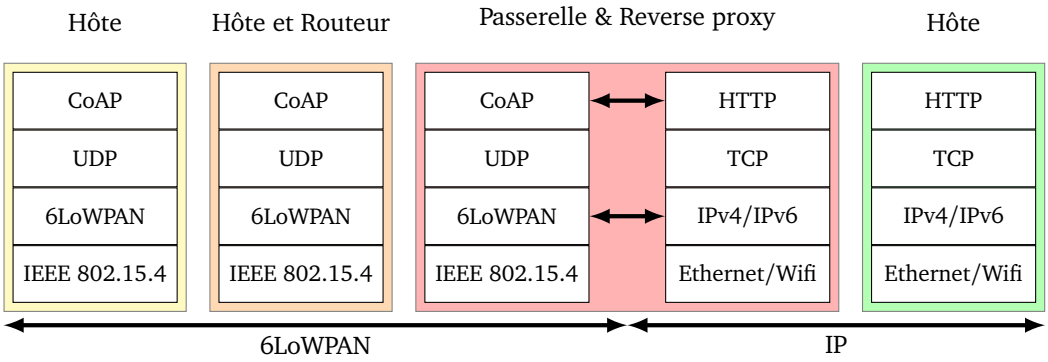


FIGURE 2.6 – Schéma de l'acheminement des requêtes dans un LLN

L'échange de messages sur un réseau LLN repose sur un ensemble de couches (au même titre qu'un réseau classique) optimisées afin de minimiser les consommations d'énergie et de bande passante. Ces économies sont faites au niveau des couches basses (2.1.1), de la compression des en-têtes (2.1.2.3), des stratégies de routages (2.2) et de la mise en cache de messages applicatifs (2.3). La Figure 2.6 propose un récapitulatif des protocoles mis en jeu lors des déploiements typiques qui ont été faits pendant cette thèse ainsi que les traductions faites au niveau de la passerelle.

Comme vu dans la section 2.4, la supervision est un aspect critique du déploiement d'un LLN afin de s'assurer qu'il fonctionne de manière normalement. Les approches actuelles reposent essentiellement sur des envois systématiques d'information de supervision qui induisent une charge croissante à mesure que le LLN grandit, notamment si la fréquence d'émission de ces messages est élevée. Le

chapitre 3 montre comment fournir un mécanisme de mesure implicite de la consommation énergétique d'un nœud en inférant l'utilisation de sa radio à partir du trafic réseau rentrant et sortant du LLN et de la topologie de routage.

Comme vu dans la sous-section 2.3.4, les Reverse Proxy Cache (RPC) permettent de réduire le trafic applicatif reçu par les nœuds capteurs en répondant à la place des nœuds du LLN. Pour cela, le Reverse Proxy Cache garde en mémoire pour un certain temps de validité les réponses des requêtes visant les nœuds. Les approches actuelles ne modulent pas ce temps de validité en fonction des objectifs de durée de vie du réseau. Le chapitre 4 montre comment adapter les temps de validité utilisés par le RPC afin d'adapter la quantité de trafic admise dans le LLN. Ce mécanisme utilise la topologie fournie par le protocole de routage du LLN et une mesure du trafic entrant pour fournir des temps de validité optimaux pour un objectif de durée de vie donné.

Chapitre 3

Mesure implicite de l'utilisation de la radio d'un LLN

> There are more things in heaven and earth, Horatio, than are dreamt of in your philosophy
>
> Shakespeare - Hamlet (1.5.167-8)

Sommaire

Ce chapitre évalue une méthode de mesure passive de l'utilisation de la radio des nœuds sans les solliciter explicitement. Cette méthode utilise de manière opportuniste les informations présentes au niveau du routeur de bordure (topologie et trafic réseau) afin de prédire celui qui est traité par les nœuds et l'utilisation de la radio induite. Cette méthode ainsi que sa précision est testée au travers de simulations et permet de conclure sur sa pertinence en fonction d'autres mécanismes présents dans le réseau.

La section 3.1 introduit les objectifs et contraintes d'une supervision dans les LLNs et justifie une approche de la supervision aussi économe en ressource que possible.

La section 3.2 décrit les approches disponibles dans l'état de l'art pour superviser des LLN et comment l'approche choisie dans ce chapitre s'en distingue.

La section 3.3 couvre la modélisation de la mesure passive utilisée en découpant l'impact selon le trafic réseau induit et d'autre part sur les temps d'activités de la radio.

La section 3.4 couvre les validations expérimentales faites pour tester la précision de la mesure implicite. Les résultats seront mis en perspectives avec de multiples métriques mesurées au cours de l'expérience afin d'expliquer le niveau de précision obtenu.

La section 3.5 montre comment améliorer la précision de la mesure implicite en la recalibrant périodiquement sur une mesure explicite. Cette section présente également les travaux en cours de recherche sur la réduction de l'impact de la mesure active.

La section 3.6 conclut ce chapitre en récapitulant les avantages et les limitations de l'approche présentée et ouvre sur d'autres contextes où les mesures implicites peuvent apporter des solutions de supervision transparentes pour les LLNs.

Introduction à la supervision d'un LLN

Comme vu dans la section 2.4, la supervision désigne la surveillance continue d'un système afin de diagnostiquer ses pannes et s'assurer que son fonctionnement est normal [115]. Un LLN met en jeu des nœuds capteurs qui mesurent ou agissent sur leur environnement. Ces nœuds ont de multiples composants (Central Processing Unit (CPU), radio, flash, ...) et ils peuvent subir des problèmes et des pannes hétérogènes et potentiellement difficiles à diagnostiquer.

Objectifs de la supervision dans les LLNs

Le but de la supervision est de permettre à un administrateur de collecter des métriques et de les visualiser afin de comprendre le fonctionnement d'un LLN pour prendre des décisions informées à son sujet.

Diagnostic de problèmes

Quand une panne ou un problème survient, l'administrateur souhaite disposer d'informations aussi pertinentes que possible pour comprendre rapidement la situation et y apporter une réponse efficace. La supervision apporte ces informations sous la forme de métriques et de mesures collectées dans le LLN et présentées à l'administrateur.

Anticipation de problèmes

Un administrateur souhaite prévenir certains problèmes prévisibles arrivant sur un LLN en anticipant l'évolution de certaines métriques. C'est notamment le cas des problèmes d'épuisement de batteries d'un LLN qui peuvent être évités en disposant d'estimations de la durée de vie des nœuds qui le composent. Ainsi un administrateur peut prévoir en avance ses opérations de maintenance et évite une panne.

Mesure des performances

Un LLN comporte plusieurs paramètres de fonctionnement (fréquence de cycle de veille, nombre d'essais d'envois de messages, etc.) qui doivent être configurés en accord les uns avec les autres pour

fournir des performances acceptables. La supervision permet de présenter des rapports de performances mettant en évidence l'impact des différents paramètres sur plusieurs métriques d'un LLN afin de le configurer de manière optimale.

Contraintes liées à la supervision d'un LLN

Superviser un LLN est important, car cela permet de garantir la fiabilité en aidant le diagnostic des pannes en apportant des mesures tout au long de son déploiement. Dans les cas classiques, la supervision se fait en envoyant des requêtes aux équipements surveillés [115]. Cependant, la supervision dans les LLNs est plus délicate, car acquérir une métrique régulièrement n'est pas toujours possible. De plus, même lorsque c'est possible, obtenir une métrique consomme des ressources, notamment lorsque cette métrique est tenue à jour régulièrement.

Contraintes liées aux nœuds

Les techniques de supervision usuelles reposent le plus souvent sur l'hypothèse qu'un nœud peut répondre aux requêtes de supervision en permanence [191]. Or ces hypothèses ne sont pas validées dans les LLNs où les nœuds sont le plus souvent endormis afin d'économiser leurs ressources.

D'autre part, un nœud n'a pas forcément un moyen programmable d'accéder à ses propres métriques pour diverses raisons (contraint en mémoire, absence de support dans le système d'exploitation utilisé, etc.).

De plus, il peut être coûteux d'implémenter l'acquisition de certaines métriques, car les ressources en mémoire et en architecture matérielle ne le permettent pas toujours. C'est notamment le cas de l'estimation de la quantité d'énergie restante dans la batterie qui alimente un nœud. Les batteries sont des systèmes chimiques complexes qui varient en fonction de la température, de la tension et de leur usage [27, 93]. Ainsi il est complexe de mesurer avec une précision satisfaisante la quantité d'énergie restante dans une batterie depuis un nœud fortement contraint et incapable de mesurer tous les paramètres requis [136].

Contraintes liées à la métrique

Certaines métriques d'un LLN comme la durée de vie nécessitent des informations récentes de la part de l'ensemble des nœuds qui compose le LLN [32]. Ainsi leur coût augmente à mesure que le nombre de nœuds dans le réseau augmente. Il est donc important de trouver des dispositifs permettant d'obtenir ces métriques en utilisant aussi peu de ressources que possible pour les obtenir.

Conclusion intermédiaire

La supervision d'un LLN peut être coûteuse en raison de la nature des nœuds, des contraintes liées à leur déploiement et de la complexité des métriques à collecter. Or cette supervision est nécessaire pour garantir la fiabilité du fonctionnement d'un LLN.

Ce chapitre propose de réduire la complexité de cette mise en place et son coût par l'utilisation de mécanismes passifs qui utilisent les propriétés connues du LLN pour déduire des métriques sans solliciter les nœuds directement.

État de l'art sur la supervision dans les LLNs

Quelques contributions ont examiné le problème général de la surveillance d'un LLN efficace en énergie. Par exemple, [116] et [104] considèrent le problème de la sélection d'un sous-ensemble

de capteurs “sondeurs” chargés de surveiller activement les autres capteurs “sondés”. Les sondeurs émettent des alarmes vers la passerelle s'ils détectent une anomalie. [116] propose un algorithme distribué d'approximation pour sélectionner un nombre minimum de sondeurs et étudie le taux de faux positif généré. [104] 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. Cependant, ces approches nécessitent des sondes déployées dans le réseau et reviennent donc à des approches actives distribuées qui peuvent être coûteuses à mettre en place.

Dans [31], les auteurs proposent LiveNet, une architecture de surveillance semi-passive qui repose sur des sondes situées dans le LLN. En utilisant les traces agrégées transmises à la passerelle, LiveNet est capable de reconstruire la topologie de routage du LLN et de déterminer divers paramètres de performance. Ce travail vise explicitement la surveillance de l'énergie, mais pourrait être adapté à d'autres indicateurs de performance. Cependant, il nécessite la transmission et le traitement de traces issues des sondes dans le réseau, ainsi il n'est donc pas entièrement pertinent pour l'approche recherchée dans ce chapitre qui n'utilise pas de sondes et ne veut pas ajouter une nouvelle charge aux nœuds.

Dans [203], les auteurs introduisent une méthode distribuée pour créer une carte de l'énergie restante d'un LLN. Les nœuds déclarent leur énergie résiduelle à un nœud voisin chargé de l'agrégation et de la compression de ces informations et ne transmettent que des mises à jour incrémentielles (condensées) à la passerelle. Suivant cette idée, [131] 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, [131] compare une méthode probabiliste, basée sur les chaînes de Markov, une méthode statistique, sur la base d'un modèle auto-régressif, avec une méthode simple de déclaration explicite pour estimer un temps de vie. Dans [82], les auteurs étendent cette idée en modélisant l'énergie de chaque nœud avec un modèle de Markov caché dont les coefficients sont estimés avec des mesures explicites. Dans [30], 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 la mesure efficace, toutes les méthodes mentionnées ci-dessus reposent fortement sur des rapports explicites et continus des nœuds et nécessitent donc une approche active alors que l'objectif de ce chapitre est d'apporter une méthode aussi passive que possible.

D'autre part, l'observation passive du trafic réseau est une pratique courante qui permet de mesurer ses principales propriétés sous la forme de flots réseau [134]. Ces flots sont spécifiés par différents protocoles tels qu'IPFIX [35] ou sFlow [146] et sont envoyés vers un collecteur qui sera ensuite interrogé par un logiciel d'analyse afin de diagnostiquer des problèmes (par exemple des congestions) ou de fournir des estimations de la quantité de flux traitée afin de procéder par exemple à une facturation. Bien qu'utilisées dans les réseaux classiques, ces techniques d'observations passives et d'analyses de flots ne sont le plus souvent pas disponibles sur les routeurs de bordure destinés aux LLNs. L'approche proposée dans ce chapitre s'inspire de cette méthodologie et propose d'utiliser ces observations de trafic pour en déduire des informations sur les nœuds du LLN.

Mesure de l'utilisation de la radio implicite et passive

L'utilisation de la radio désigne le temps passé par chaque nœud dans les différents états d'utilisation de sa radio (transmission, réception, écoute du canal, etc.). C'est une des métriques les plus importantes d'un nœud dans un LLN car elle permet de détecter des conditions difficiles de transmissions (grand nombre de retransmissions, environnement dense, etc.).

De plus, elle permet de déduire une estimation de la consommation énergétique d'un nœud cap-

teur, ce qui est crucial pour disposer d'une estimation de sa durée de vie. Cette déduction est obtenue en multipliant le temps passé dans chaque état par sa puissance nominale afin d'obtenir une estimation de la consommation d'énergie. Dans les déploiements usuels, la radio est la première source de consommation énergétique. Ainsi, prévoir l'utilisation de la radio permet de prévoir une large part de la consommation énergétique d'un nœud [132].

La mesure passive de l'utilisation de la radio d'un nœud permet de mesurer implicitement cette grandeur sans la demander explicitement afin d'économiser de l'énergie d'une part et d'autre part d'avoir un mécanisme de supervision fonctionnant sans faire d'hypothèses au sujet des capacités d'un nœud.

Le reste de ce chapitre se concentre sur les moyens d'obtenir une estimation passive de l'utilisation de la radio. L'objectif est de mesurer cette grandeur en la déduisant à partir d'informations connues et en ne sollicitant pas les nœuds capteurs afin d'économiser leur énergie.

Architecture de mesure implicite

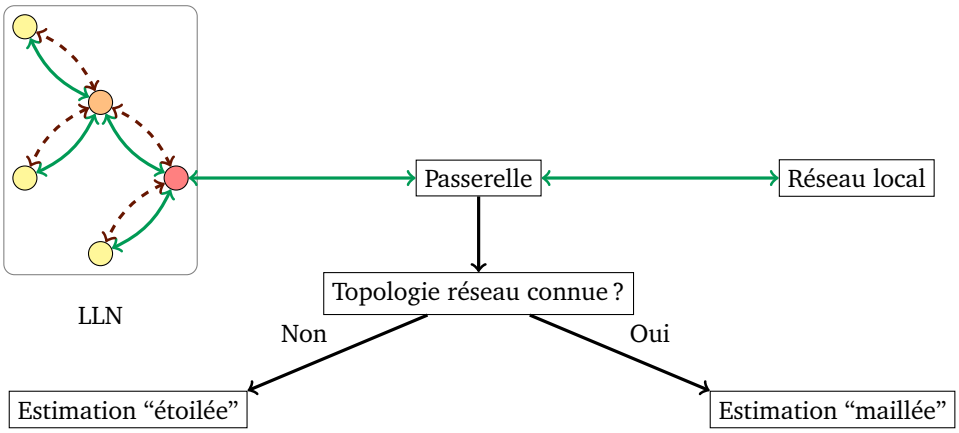


FIGURE 3.1 – Architecture de la mesure implicite du trafic réseau

La Figure 3.1 illustre l'architecture choisie pour estimer l'utilisation de la radio de chaque nœud. Cette architecture met à profit le trafic passant par le routeur de bordure afin d'en déduire des informations sur l'utilisation de la radio dans les LLNs. La passerelle agit comme un routeur de bordure et peut ainsi voir le trafic montant et descendant du LLN qui est représenté en vert sur la figure. Ces traces de trafic sont agrégées afin de fournir pour chaque nœud son trafic montant et son trafic descendant.

Deux scénarios peuvent ensuite arriver : soit la topologie de routage est connue au niveau du routeur de bordure soit elle ne l'est pas.

Dans le cas où la topologie est inconnue [1], l'estimation du trafic ne comptabilisera que la source et la destination d'un message. Le routeur de bordure alors n'a pas de connaissance sur la topologie du LLN et ne peut donc fournir qu'une estimation à un saut. L'estimation est dite "étoilée" car elle correspond à une estimation sur une topologie étoilée où chaque nœud serait connecté à un saut avec la passerelle.

Dans le cas où la topologie des routes est connue [2], le routeur de bordure peut déduire pour chaque paquet, les routeurs intermédiaires traversés dans le LLN et donc les nœuds impliqués dans

1. C'est par exemple le cas quand RPL fonctionne en "storing mode" [197].
2. C'est par exemple le cas pour les routes descendantes quand RPL fonctionne en "non-storing" mode [197].

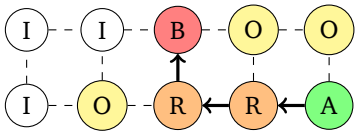


FIGURE 3.2 – Nœuds impactés par l'acheminement de paquets depuis le nœud A vers le nœud B.

la transmission d'un paquet. Le routeur de bordure alors a une connaissance sur la topologie du LLN et fournit alors une estimation "maillée" correspondant à la topologie multi sauts connue.

Dans ces deux cas de figures, une estimation de l'impact du trafic réseau sur l'ensemble des nœuds qui composent le LLNs est fournie. Cependant, dans le cas d'une estimation "étoilée", l'impact sur les relais ne sera pas estimé.

La taille des paquets étant connue, de même que la couche radio, il est alors possible de déduire combien de temps un nœud utilise sa radio pour transmettre un paquet et respectivement pour le recevoir. Donc, la passerelle peut calculer une estimation du temps passé par un nœud en transmission et réception en ne demandant rien aux nœuds et en ne consommant ainsi aucune énergie.

Cependant, ce mécanisme de supervision est biaisé, il ne peut pas prendre en compte les phénomènes locaux qui ne passent pas par la passerelle, ce qui conduit à des prévisions sous-estimées. Cette partie du trafic (perte de paquets, transmissions locales, etc.) est inconnue du routeur de bordure et est représentée sur la Figure 3.1 en pointillé marron. L'objectif est donc d'évaluer la précision que l'on peut attendre de telles estimations en fonction de la configuration d'un LLN.

La passerelle ne peut connaître l'ensemble du trafic et donc ne peut remplacer la précision d'une mesure active, mais peut offrir une alternative lorsqu'elle n'est pas envisageable en raison des diverses contraintes d'un déploiement.

Modélisation de l'impact du trafic réseau

Un nœud peut être le destinataire d'un paquet réseau, mais peut aussi être utilisé comme un routeur intermédiaire dans un chemin multi-sauts.

La Figure 3.2 représente une transmission d'un paquet entre un point A et B dans un LLN typique et permet d'illustrer les nœuds impliqués dans un réseau maillé pour le routage d'un paquet. Les liens en pointillés représentent les liens radios entre les différents nœuds et les nœuds représentés par un R sont les routeurs intermédiaires pour acheminer les paquets. Les nœuds représentés par un O peuvent être sujets à des phénomènes d'Overhearing tandis que les nœuds représentés par un I ne sont pas affectés par cette transmission.

Le routeur de bordure peut observer les entêtes des paquets réseau qu'il route et extraire les sources et destinations pour chacun de ces paquets. Soit $\mathscr{S}_i$ les paquets ayant le nœud i pour source et $\mathscr{D}_i$ les paquets ayant pour destination le nœud i. Le routeur de bordure a une vue sur $\mathscr{S}_i$ et $\mathscr{D}_i$ pour chaque nœud dans le LLN.

En outre, pour chaque paquet, le routeur connaît également sa taille et peut donc en connaissant la couche physique utilisée connaître le temps passé par un paquet sur le médium. De plus, une couche liaison peut disposer d'un mécanisme d'acquittement par l'intermédiaire d'un paquet `ACK` envoyé par le destinataire lorsqu'une transmission a fonctionné. Soit T_p le temps pris par un paquet pour être transmis et $T_{\texttt{ACK}}$ le temps nécessaire pour transmettre un `ACK` au niveau de la couche liaison.

Mesure passive "étoilée"

Le routeur de bordure n'a pas forcément accès à la topologie d'un LLN, ce qui peut par exemple arriver quand RPL est utilisé en "storing-mode". Dans ce cas, le routeur de bordure ne peut voir que

la source et la destination dans l'entête d'un paquet qu'il doit router et agit comme si la topologie du réseau était étoilée. Dans le scénario présenté sur la figure. 3.2, l'estimation "étoilée" ne déduit que les temps de radios utilisés par les nœuds A et B en ignorant les autres nœuds impliqués.

$$\mathrm{Tx}_i = \sum_{p \in \mathscr{S}_i} T_p + \sum_{p \in \mathscr{D}_i} T_{\texttt{ACK}} \tag{3.1}$$

$$\mathrm{Rx}_i = \sum_{p \in \mathscr{D}_i} T_p + \sum_{p \in \mathscr{S}_i} T_{\texttt{ACK}} \tag{3.2}$$

Ce type de supervision peut être adapté pour des topologies étoilées comme celles vues dans la section 2.1.1.1. Cependant, dans le cas des scénarios maillés, cette méthode de mesure sous-estimera l'impact du routage intermédiaire et doit être complétée avec une connaissance de la topologie.

Mesure passive "maillée"

Un routeur de bordure peut avoir la connaissance des routes descendantes vers les nœuds, c'est par exemple le cas quand RPL est utilisé en "non-storing mode". Dans ce cas, les routes descendantes sont utilisées pour inférer l'impact d'un paquet sur le LLN de manière plus fine. De plus, dans le cas où un nœud est configuré pour n'utiliser qu'un seul parent à la fois, la topologie des routes montantes est équivalente à celle descendante.

Dans le reste de cette thèse, il sera supposé que chaque nœud utilise exclusivement son parent préféré pour les routes montantes.

Ainsi lorsque les informations sur le routage sont disponibles il est possible de compléter la supervision passive pour prendre en compte l'impact causé par le relayage des paquets. Dans le scénario présenté sur la figure. 3.2, l'estimation "maillée" comptabilise les temps de radios utilisés par les nœuds A et B mais aussi ceux des nœuds routeurs (R).

Soit $\mathscr{F}_i$ l'ensemble des paquets qui sont routés par le nœud i alors qu'il n'est ni la source ni le destinataire du paquet.

$$\mathrm{Tx}_i = \sum_{p \in \mathscr{S}_i \cup \mathscr{F}_i} T_p + \sum_{p \in \mathscr{D}_i} T_{\texttt{ACK}} \tag{3.3}$$

$$\mathrm{Rx}_i = \sum_{p \in \mathscr{D}_i \cup \mathscr{F}_i} T_p + \sum_{p \in \mathscr{S}_i(t)} T_{\texttt{ACK}} \tag{3.4}$$

Remarque sur l'"over-hearing"

Les nœuds ne peuvent pas savoir si une trame leur est destinée avant de la recevoir ainsi ils peuvent dépenser de l'énergie pour décoder une trame sans qu'elle ne leur soit destinée ("over-hearing" en anglais). Donc, la connaissance des routes n'est pas encore suffisante pour prévoir intégralement l'impact du trafic sur un LLN, la supervision passive idéale prendrait également en compte les phénomènes d'over-hearing.

Il est possible d'envoyer ces informations en utilisant des paquets déjà existants (routage ou applicatif) et ainsi d'avoir à la passerelle ces informations par un mécanisme de "piggy-backing". Ou bien, d'utiliser la connaissance des voisins qui peut être donnée par un administrateur dans le cas où les emplacements sont connus.

Cependant, la qualité du lien radio peut changer au cours du temps et la passerelle ne peut pas l'inférer sans rien demander aux nœuds. Ainsi ce phénomène ne sera pas pris en compte lors de l'estimation passive de l'usage de la radio des nœuds, car aucune hypothèse n'est faite sur les informations que les nœuds envoient.

Modélisation de l'utilisation de la radio d'un nœud

La passerelle du fait de sa connexion avec les nœuds connaît la technologie sans-fil qui va être utilisée pour communiquer et peut donc fournir une modélisation de son impact.

L'interface radio est la principale consommatrice d'énergie dans la plupart des LLN [10]. En particulier, l'énergie consommée par la transmission et la réception sont du même ordre [52]. Une utilisation importante de la radio implique dans les deux cas une grande quantité d'énergie consommée par le nœud.

L'objectif est de calculer le temps passé par chaque nœud pour transmettre un paquet p de taille $\mathscr{L}(p)$ octets envoyé en unicast et avec un acquittement de taille constante. Les modélisations suivantes se concentrent sur la radio IEEE 802.15.4, de plus, afin d'économiser de l'énergie, il est également pris pour hypothèse que les nœuds utilisent un mécanisme de cycle de veille. ContikiMAC [54] dispose d'un cycle d'endormissement très économe en énergie [129]. Il sera modélisé et utilisé par les nœuds.

Impact de IEEE 802.15.4

IEEE 802.15.4 peut être utilisé de plusieurs façons différentes (taille courte ou longue d'adresse, avec ou sans couche MAC, etc.) [19]. Cette section prend pour hypothèse que IEEE 802.15.4 fonctionne avec le protocole Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) et que le temps utilisé par un paquet individuel p sur la radio est donné de la manière suivante :

$$T_p = \frac{\mathscr{L}(f)}{R}\left\lceil \frac{\mathscr{L}(p)}{L} \right\rceil$$

où $\mathscr{L}(f)$ est la taille d'une trame de donnée complète IEEE 802.15.4 (127 octets), $R = 250$ kbits/s est le débit du IEEE 802.15.4, L est la charge utile (payload) maximale d'une trame de donnée IEEE 802.15.4 (environ 89 octets avec les options usuelles) et $\mathscr{L}(p)$ la taille du message que l'on veut transmettre sur ce médium. Une trame de donnée complète prend environ 4 ms pour être transmise.

Une trame d'acquittement `ACK` en IEEE 802.15.4 est de taille constante (11 octets) et occupe donc la radio pendant $T_{\texttt{ACK}} = 0.352$ ms.

Quand une source émet une trame, tous les nœuds en écoute qui sont à portée radio 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 leur est destinée. 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.

Enfin, il est à noter que même quand aucun paquet n'est transmis, les nœuds utilisent leur radio afin de vérifier l'état du canal dans le but de recevoir un paquet. Cependant, ce réveil ne peut être déduit implicitement de la part de la passerelle et rentre donc dans les phénomènes non couverts par la mesure passive.

Impact de ContikiMAC

Afin de communiquer, deux nœuds ont besoin d'être actifs en même temps. Cependant, mettre les nœuds en réception permanente est très coûteux car une radio en écoute du canal ("channel listening") consomme une énergie non négligeable [140]. Ainsi afin d'économiser les réserves d'énergie des nœuds, les nœuds s'endorment régulièrement en utilisant des mécanismes de cycle de veille qui alternent phase de sommeil et phase d'activité.

L'objectif de ContikiMAC [54] consiste à endormir les nœuds pendant une partie du temps afin d'économiser leur batterie et d'éviter les collisions dans les zones denses où plusieurs nœuds veulent transmettre simultanément. Ce mécanisme de cycle de veille est notamment utilisé par Contiki [53]

en raison de son efficacité à économiser de l'énergie et sa faible empreinte mémoire [129]. Ainsi il sera utilisé comme référence de cycle de veille dans ce chapitre.

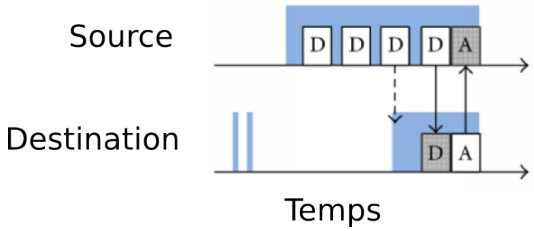


FIGURE 3.3 – Fonctionnement des réceptions avec ContikiMAC

La Figure 3.3 illustre le fonctionnement de ContikiMAC. Pour transmettre une trame de donnée (D), un émetteur l'envoie de manière répétée ("packet strobing" en anglais) tant qu'une trame d'acquittement (A) n'est pas reçue de la part du récepteur. Un récepteur va se réveiller périodiquement et faire une écoute du canal pour détecter la présence d'une trame (l'écoute du canal est représentée en bleu sur la figure). Si aucune émission n'est détectée, le récepteur se rendort. Quand une émission est détectée, le récepteur passe en réception pour recevoir la trame de donnée dans son intégralité puis, quand la trame de donnée est reçue, une trame d'acquittement est envoyée du récepteur vers l'expéditeur.

Dans le cas d'une trame destinée à tous les voisins (broadcast), elle est répétée sur toute la phase d'activité afin de s'assurer que tous les voisins l'ont reçue au moins une fois. D'autre part, les voisins n'envoient pas un acquittement à la réception d'une trame envoyée en broadcast.

Si une trame est transmise sur le canal pendant la phase de réveil, le récepteur reste réveillé pour la recevoir sinon il se remet à l'état sommeil. Lorsqu'un nœud reçoit une trame et qu'elle lui est effectivement destinée, il envoie un acquittement.

Pour optimiser sa transmission, le protocole ContikiMAC utilise une technique de verrouillage de phase. Elle consiste à estimer la date de réveil du prochain saut pour une destination donnée à partir de la date de réception d'acquittement et de retarder le moment d'émission de paquets jusqu'à l'approche de cette date de réveil. Cette technique peut économiser de l'énergie en réduisant la quantité de strobing nécessaire à l'envoi d'un paquet en "synchronisant" envois et phases de réveil. Cependant, cette technique n'est pas parfaite, car les horloges internes des différents nœuds ne sont pas synchronisées et des décalages peuvent tout de même se produire [69].

Toujours dans l'objectif d'augmenter la durée de la période de sommeil, le protocole ContikiMAC dispose d'un mécanisme de mise en sommeil rapide pouvant distinguer une transmission de paquet du bruit en comparant la période de l'activité à des longueurs types : (supérieure à la longueur maximale d'un paquet, etc.). Si la période d'activité n'est pas due à une transmission, le nœud se rendort immédiatement.

Ainsi ContikiMAC permet d'économiser de l'énergie par de multiples techniques, mais cette économie se paye par l'envoi répété d'une trame. Donc, il est nécessaire de comptabiliser ces retransmissions afin d'avoir une mesure implicite pertinente.

Soit N_{sender} (respectivement N_{receiver}) le nombre de fois qu'une source (respectivement une destination) va transmettre (respectivement recevoir) un paquet avec ContikiMAC.

Puisque le destinataire se réveille en moyenne au milieu d'une transmission et attend le prochain essai d'envoi pour recevoir la trame complètement et envoyer son acquittement, le nombre de trames reçues par le destinataire peut être estimé à $N_{\text{receiver}} = 1.5$.

Estimer N_{sender} est plus difficile, car cette grandeur dépend des conditions réseau (topologie, qualité du lien) et de la taille des trames. Ainsi il est nécessaire d'avoir une estimation de ce paramètre

afin de procéder à une estimation réaliste de l'utilisation de la radio.

L'expérience suivante est réalisée afin d'estimer le nombre moyen d'essais d'envoi d'une trame de donnée qu'un expéditeur va faire avant de recevoir une trame de confirmation. Deux nœuds utilisant ContikiMAC sont mis à portée de transmission dans un environnement non bruité. La source envoie en boucle un paquet de taille fixe qui induit de multiples retransmissions, puis elle attend l'acquittement du paquet et comptabilise le nombre d'essais réalisés avant de recevoir un acquittement et cela pour différentes tailles de paquets. IEEE 802.15.4 est utilisé comme simple couche liaison avec un entête pour les trames de données aussi réduit que possible (6 octets) les différentes tailles de trames sont obtenues en faisant varier la taille du message applicatif. Une simulation simule le comportement de ce système pendant 10000 secondes et pour chaque taille de paquet, l'expérience est exécutée 10 fois avec une graine de générateur de nombre aléatoire différente à chaque fois.

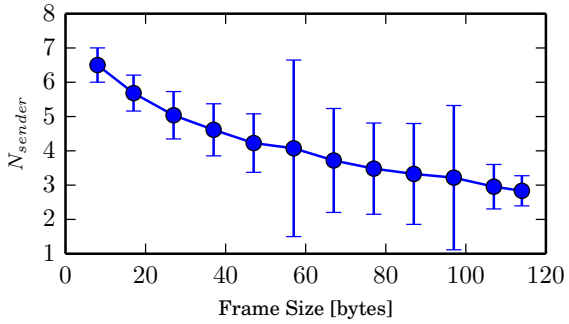


FIGURE 3.4 – Nombre moyen de tentatives d'envois en fonction de la taille de la trame avec ContikiMAC.

La Figure 3.4 représente la moyenne du nombre de transmissions $N_{\text{sender}}(f)$ nécessaires à l'envoi d'une trame f ayant une taille de $\mathscr{L}(f)$ octets entre deux nœuds fonctionnant avec ContikiMAC. Le nombre d'envois moyens nécessaires pour envoyer un long paquet est plus faible que pour en envoyer un court, car les trames longues mettent plus de temps pour être transmises et ont donc plus de chance d'être écoutées pendant un réveil du destinataire.

On peut remarquer que la variance est également faible pour des trames de grande taille ce qui est également cohérent, car le nombre d'envois qu'il est possible de faire pour une trame de cette taille au cours d'un cycle complet est limité ainsi la variance est réduite.

Validation expérimentale

Supervision passive

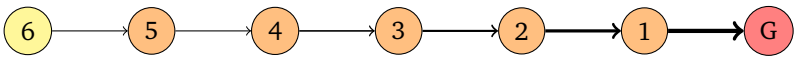


FIGURE 3.5 – Topologie réseau utilisée pour mesurer la précision de la mesure implicite et passive

Une topologie chaînée à 7 nœuds comme représentée sur la Figure 3.5 est utilisée pour tester le mécanisme décrit ci-dessus. Dans cette configuration, les nœuds ne peuvent envoyer et recevoir de paquets que de la part de leurs voisins adjacents. Tous les paquets remontent vers la passerelle représentée par un G sur la figure.

Les nœuds utilisent le système Contiki [53], IEEE 802.15.4 pour communiquer et ContikiMAC comme mécanisme de cycle de veille. RPL [197] est le protocole de routage utilisé et il est configuré

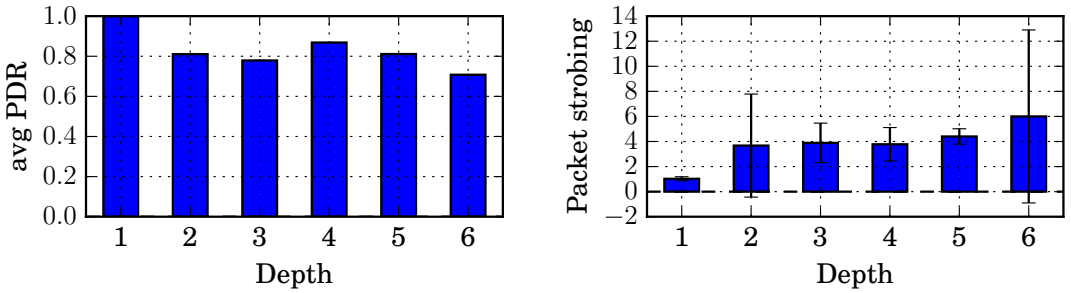
en non-storing mode, c'est-à-dire qu'il a accès à toute la topologie réseau et en particulier aux routes descendantes.

Durant 200 secondes, chaque nœud envoie au LBR un paquet UDP à chaque seconde avec une charge utile de 10 octets ce qui induit une trame de 69 octets chaque seconde. Le trafic applicatif d'un nœud démarre lorsqu'il joint la topologie réseau, il a alors un parent qui est son voisin adjacent.

D'après les résultats expérimentaux (Figure. 3.4), l'utilisation de ContikiMAC induit qu'une trame de 69 octets est émise en moyenne 3,76 fois avant d'être reçue. Le mécanisme de mesure implicite tient compte de ce phénomène et l'applique à chaque routeur intermédiaire afin que l'utilisation de la radio soit en accord avec ce résultat.

Les temps de transmission et de réception des nœuds sont obtenus par l'utilisation du simulateur et de Powertracker [55] qui permet d'obtenir le temps passé dans chaque état de transmission radio pour chaque nœud avec une résolution de $1\mu s$. Puisque la passerelle est toujours éveillée, elle reçoit donc du premier coup toutes les trames : $N_{\text{receiver}} = 1$.

Analyse de l'impact de la profondeur



(a) Packet Delivery Ratio (PDR).

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

FIGURE 3.6 – Impact de la profondeur

Packet Delivery Ratio (PDR)

La figure Fig. 3.6a représente le ratio de succès de transmission de paquets applicatifs pour chaque nœud en fonction de la distance à la racine en nombre de sauts. Cette figure montre la proportion du trafic applicatif que la passerelle peut effectivement voir. Le ratio de paquets acheminés n'est pas uniforme sur tout le LLN et les nœuds qui ne sont pas connectés directement à une racine souffrent de pertes de paquets non négligeables malgré un trafic faible en raison d'un grand nombre de retransmissions de paquets.

Le routeur de bordure n'utilise pas ContikiMAC car dans les hypothèses de l'expérience, il n'est pas contraint en énergie et peut donc rester éveillé en permanence. Ainsi, tous les paquets issus du nœud 1 sont reçus prouvant que ContikiMAC induit des pertes et nuit à la fiabilité des transmissions pour les nœuds plus éloignés.

Puisqu'un paquet peut être perdu à chaque fois qu'il traverse un routeur fonctionnant avec ContikiMAC, traverser de nombreux routeurs augmente la probabilité que les paquets soient perdus, car UDP ne fournit aucun mécanisme de fiabilité. Ce phénomène est confirmé par la Figure 3.6a où la quantité de paquets arrivant à la racine semble décroître globalement à mesure qu'on s'en éloigne.

Retransmissions de paquets (“Packet strobing”)

La Figure 3.6b représente le nombre moyen d’envois de paquets i.e. le nombre de tentatives que va faire chaque nœud pour envoyer une trame à son parent.

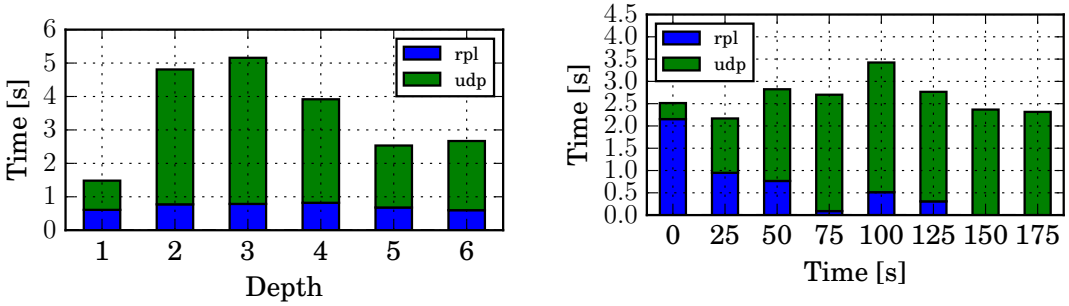
Les valeurs obtenues précédemment restent expérimentalement proches de la valeur de 3,76 obtenue expérimentalement (Figure 3.4). Cependant, la variance est forte et le nombre de tentatives tend à augmenter à mesure que l’on s’éloigne de la racine. Cette variance peut être expliquée par la variance obtenue pour des trames de 69 octets sur l’expérience de calcul des N_{sender}.

La convergence vers la valeur normale de 3,76 pour les nœuds proches de la racine s’explique par l’augmentation du trafic réseau sur ces nœuds : ils doivent relayer les paquets provenant de tous les autres nœuds. Ainsi, leur rythme d’envoi se rapproche de celui de l’expérience de strobing où la source envoyait autant de paquets que possible vers un destinataire en s’assurant de son acquittement. Pour un nœud situé en bout de chaîne et n’ayant donc aucune charge de routage, un plus grand décalage entre les cycles d’éveils est présent et explique ainsi les retransmissions supplémentaires.

Enfin, au sujet du nœud qui est connecté à la passerelle directement, la quantité moyenne d’envois est très proche de 1 avec une variance faible, ce qui est logique puisque par hypothèse la racine est toujours éveillée et peut donc recevoir immédiatement les transmissions qui en viennent. Cependant, ce n’est pas le cas des nœuds se situant juste avant le nœud 1, qui reçoivent une quantité importante de trafic, mais qui ne peuvent pas la router avec la même efficacité (temporisation, “buffering”, etc.).

Analyse de l’impact des protocoles

Connaître la répartition des protocoles au cours du temps permet de savoir la proportion de ressources investies dans le maintien du réseau et celle investie dans un but applicatif à valeur ajoutée.



(a) Répartition des protocoles par profondeur.

(b) Évolution de la répartition des protocoles (statistiques par fenêtres de 25 secondes.

FIGURE 3.7 – Impact des protocoles

Répartition des protocoles

La Figure 3.7a met en évidence que la quantité de trafic lié au protocole de routage est relativement homogène pour tous les nœuds. Ce phénomène est lié à la topologie et au fonctionnement de RPL : les nœuds n’envoient de paquets de routage qu’à un seul parent à la fois ainsi le trafic de routage est le même pour tous les nœuds du réseau.

Évolution au cours du temps

La Figure 3.7b met en évidence l'évolution typique de la répartition des protocoles au cours du temps. La charge liée au protocole de routage est très importante lors du démarrage du réseau, car aucune route n'est alors disponible. Ainsi l'essentiel du trafic observé correspond à des paquets RPL qui sont utilisés pour mettre à jour les routes sur chaque nœud.

RPL étant proactif, les routes sont construites et entretenues en permanence, ainsi même après la construction de la topologie, du trafic RPL est toujours émis. Cependant, le mécanisme de "Trickle" réduit la quantité de paquets émis pour maintenir la topologie qui est observée.

Cette évolution du trafic permet de déduire qu'un mécanisme n'ayant pas la connaissance des routes manquerait non seulement l'impact du relayage sur les routeurs dans la topologie, mais aussi la charge impliquée par la construction des routes qui est non négligeable même dans le cas d'une topologie simple comme celle d'une chaîne.

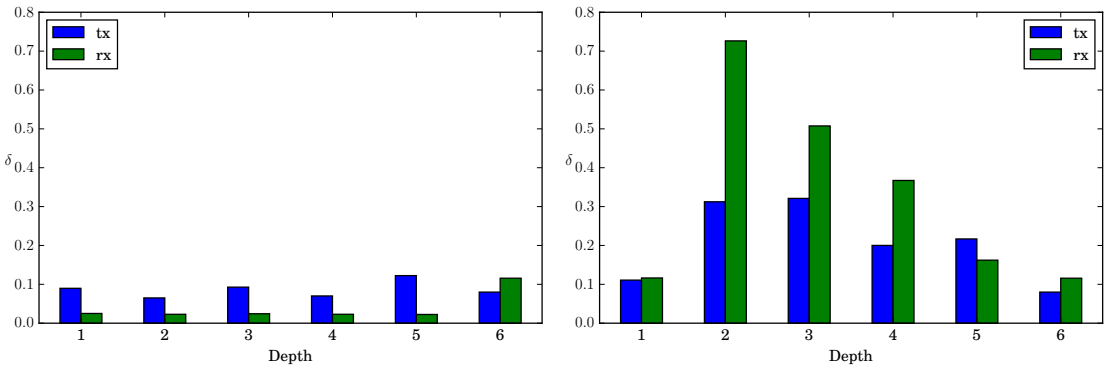
Cependant, le trafic applicatif est majoritaire lorsque le réseau est stable, ainsi l'approche par supervision passive peut être explorée, car l'essentiel du trafic a vocation à passer par le routeur de bordure.

Précision de la mesure passive de l'utilisation de la radio

Le mécanisme d'estimation se déclenche quand toutes les routes sont établies et qu'un trafic réseau est observé à la passerelle. L'estimation du temps radio va alors fournir une estimation pour le temps passé en transmission et en réception pour chaque nœud.

On appelle *précision* $\delta = \frac{\widehat{X}}{X}$ le rapport entre l'estimation d'une grandeur $\widehat{X}$ et sa valeur réelle X qui est mesurée dans le simulateur. Cette métrique permet d'interpréter facilement les grandeurs trouvées : $\delta \leq 1$ (respectivement $\delta \geq 1$) signifie que l'estimation sous-estime (respectivement surestime) la grandeur. Cette section mesure la précision des estimations passives pour les cas d'estimation "étoilée" (où la topologie de routage est inconnue) et "maillée" (où la topologie de routage est connue).

Précision en fonction de la topologie



(a) δ total global pour l'estimation "étoilée" (b) δ total global pour l'estimation "maillée"

FIGURE 3.8 – δ total et global par profondeur

La Figure 3.8 montre que la précision obtenue à la fin de la simulation pour chaque nœud est sous-estimée. Ce phénomène est attendu, car la mesure implicite ne peut inférer tous les phénomènes, de plus l'approche proposée n'introduit pas de mécanismes ad hoc pour augmenter les estimations.

Dans le cas de l'estimation "étoilée", δ est homogène pour l'ensemble des nœuds qui ont pour charge de router des paquets qui ne leur sont pas destinés comme le montre la Figure 3.8a. C'est cohérent avec la nature de l'estimation "étoilée" qui ne prend pas en compte toute la charge liée au routage. De plus, la réception est plus sous-évaluée que la transmission, car elle ne prend en compte que la réception des trames d'acquittement qui représentent un volume très restreint sauf pour le nœud 6 qui n'a pas d'autre charge de réception.

Dans le cas de l'estimation "maillée", δ est plus élevé, car l'utilisation de la radio liée aux retransmissions est prise en charge ainsi une plus grande part du trafic est déduite des observations passives. On remarque un net progrès pour la réception qui d'une part est régulière à prévoir avec ContikiMAC ($N_{receiver} = 1.5$) et d'autre part parce que le volume de routage est plus important.

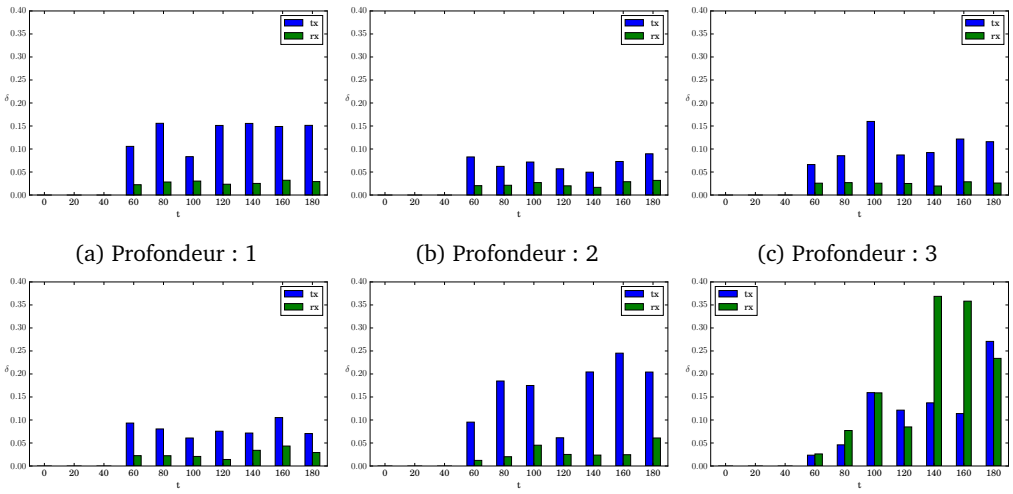
Les erreurs d'estimations restent cependant importantes, car dans les deux cas, une part importante du trafic n'est pas inféré (pertes de paquets, variance du strobbing, trafic de routage local non transmis à la passerelle, etc.).

En outre, la mesure implicite "étoilée" ou "maillée" est identique pour le nœud feuille par construction, car aucune charge de routage ne lui est appliqué dans les deux cas.

Évolution de δ au cours du temps

Cette section étudie l'évolution de δ au cours du temps afin d'affiner la compréhension des facteurs pouvant expliquer la sous-estimation obtenue dans la section précédente. Les résultats suivants sont obtenus en découpant l'expérience sur des intervalles de 20 secondes. δ est alors le temps de radio estimé sur le temps réel dans cet intervalle uniquement.

Estimation "étoilée"



(a) Profondeur : 1 (b) Profondeur : 2 (c) Profondeur : 3

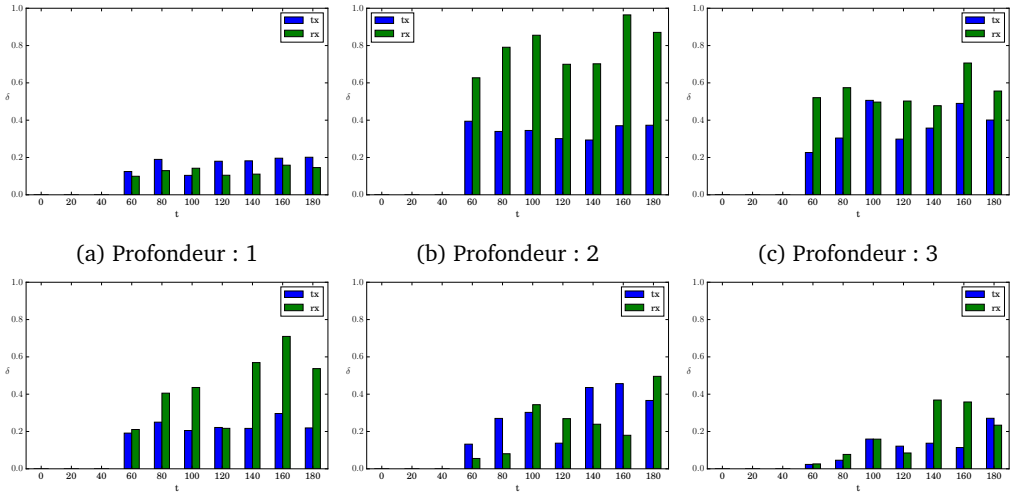
(d) Profondeur : 4 (e) Profondeur : 5 (f) Profondeur : 6

FIGURE 3.9 – Évolution de δ pour une estimation "étoilée"

La Figure 3.9 détaille pour chaque nœud l'évolution de δ pour une estimation "étoilée". La sous-estimation continue, car les processus non visible depuis la passerelle (routage des paquets, pertes de paquets, etc.) continuent au cours de l'expérience.

Les nœuds situés à une plus grande profondeur relaient moins de paquets ainsi leur δ augmente, car une plus grande proportion de leur trafic applicatif est connue par le routeur de bordure au fur et à mesure de l'expérience. D'autre part, δ augmente au cours du temps, car le trafic applicatif qui est celui inféré par la mesure passive prend une part plus importante au cours du temps (3.7b).

Estimation "maillée"



(a) Profondeur : 1 (b) Profondeur : 2 (c) Profondeur : 3

(d) Profondeur : 4 (e) Profondeur : 5 (f) Profondeur : 6

FIGURE 3.10 – Évolution de l'δ pour une estimation "maillée"

L'estimation maillée est meilleure sur tous les intervalles de temps, ce qui est attendu, car elle comptabilise la charge de routage sur chacun de ces intervalles en plus de l'estimation étoilée.

D'autre part, l'estimation s'améliore au cours du temps, car non seulement le trafic applicatif devient majoritaire au cours du temps comme montré sur la Figure 3.7b mais en plus la charge de routage intermédiaire qu'il induit est également comptabilisée.

Cependant, l'erreur d'estimation est encore forte en raison des pertes de paquets et des variations de la quantité de "strobbing" nécessaire pour transmettre un paquet à chaque saut. Ainsi si la quantité de strobbing est sous-estimée, sur chaque saut l'impact de chaque paquet sur l'estimation de la radio est grandement perturbé.

Afin de limiter le strobbing, une solution serait d'avoir des trames aussi remplies que possible (par exemple avec des informations au sujet des nœuds.). Ainsi, la quantité de strobbing nécessaire serait beaucoup plus faible, car les trames seraient détectées au bout d'un faible nombre de tentatives.

Une autre solution possible consiste à envoyer à intervalles réguliers des informations de "recalibrations" afin de recaler les estimations pour fournir des valeurs correctes.

Mesures explicites

La passerelle peut effectuer une mesure passive du temps passé par les nœuds du LLN en transmission et en réception. Cependant, les paquets perdus, retransmis ou non routés vers la passerelle ne sont pas comptabilisés avec cette méthode qui induit donc un biais.

Ainsi des mécanismes de mesure active semblent inévitables pour obtenir une information fiable. Ces messages peuvent être envoyés à la passerelle par “piggybacking” ou être mis dans les paquets du protocole de routage utilisé.

Cependant, le rythme de ces mesures doit être modéré afin de limiter la consommation de ressources qu'il induit. Ainsi si les temps entre deux envois de mesures explicites sont longs, la mesure implicite de la radio précédemment introduite peut fournir une approximation entre ces deux envois.

Quand la passerelle reçoit une mesure explicite au temps t elle recalcule son estimation pour la grandeur $\widehat{X}$ de la manière suivante :

$$\widehat{X}(t) = X(t_r) + \epsilon(t_r)\frac{t - t_r}{T} \tag{3.5}$$

$$\epsilon(t_r) = \alpha(X(t_r) - \widehat{X}(t_r)) + (1-\alpha)\epsilon(t_{r-1}) \tag{3.6}$$

où t_r et t_{r-1} sont les deux dernières dates de réception de mesure active. T désigne le temps entre chaque mesure explicite reçue. $X(t_r)$ désigne la valeur obtenue par mesure explicite à l'instant t_r. $\epsilon(t_r)$ est l'estimation de l'erreur apprise des précédents messages de mesure active[3]. Cette erreur est calculée par une Exponentially Weighted Moving Average (EWMA) de paramètre α [127]. Le coefficient α compris entre 0 et 1 représente le degré de décroissance des erreurs des valeurs passées. Une valeur de α importante réduira plus rapidement l'importance des observations passées. A un instant t, l'erreur est prise proportionnellement au temps depuis la dernière mesure explicite reçue.

Validation expérimentale

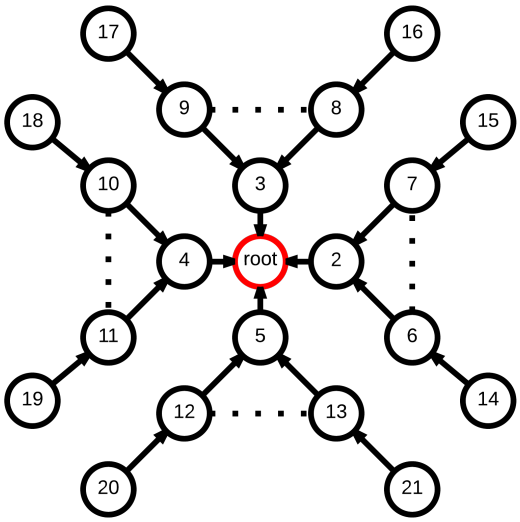


FIGURE 3.11 – Topologie réseau et radio.

Cette méthode de recalibration des mesures passives est évaluée en utilisant une topologie représentée sur la Figure 3.11 composée de 21 nœuds envoyant des paquets applicatifs à la racine chaque seconde pendant 200 secondes. Les messages contenant des mesures exactes des temps mesurés par

3. Par convention : $\epsilon(0) = 0$

le nœud passé dans chaque état de transmission sont envoyés par les nœuds par un mécanisme de publish-subscribe pour économiser le coût d'une requête. La fréquence de ces envois de recalibrations est réglée par l'administrateur.

Comme vu dans les validations précédentes, le réseau est peu régulier et les phénomènes de strobing ont une grande variance. Ainsi, afin de ne pas réagir trop brutalement aux changements d'une mesure et de privilégier les tendances de fond, le paramètre de décroissance de l'EWMA est mis à $\alpha = 0.25$.

Erreur relative globale

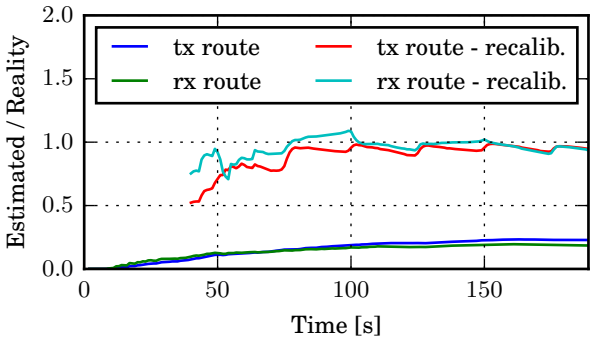


FIGURE 3.12 – Erreur d'estimation globale pour une topologie en arbre avec et sans recalibration dynamique (toutes les 25 secondes).

La Figure 3.12 montre le rapport entre la valeur estimée et la valeur réelle moyenne quand des messages de supervisions périodiques sont envoyés par les nœuds toutes les 25 secondes. A chaque mesure explicite, l'estimateur intègre l'erreur moyenne des précédentes mesures et converge ainsi vers une estimation correcte. On observe également que les pentes des estimations après chaque recalibrations sont moins grandes, car l'estimateur apprend son biais qui est supposé constant pendant l'expérience. Ainsi on obtient une estimation qui a tout temps est capable de produire une estimation du temps passé en transmission et en réception.

Fréquence de recalibrage

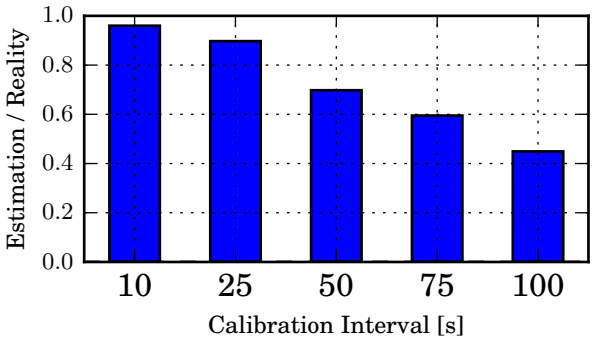


FIGURE 3.13 – Erreur relative pour différents intervalles de supervision active.

La Figure 3.13 montre l'erreur moyenne globale obtenue par le mécanisme avec une estimation étoilée en fonction des périodes entre chaque message de supervision active. Sur une expérience durant 200 secondes, n'avoir qu'une seule recalibration à mi-parcours (100 secondes) n'est pas suffisant pour avoir une bonne précision. Ce phénomène est attendu, car même si la mesure active permet de connaître l'état exact d'un nœud, n'avoir cette information qu'une fois au cours de l'expérience n'est pas suffisant. Lorsque les recalibrations sont plus fréquentes, la précision est meilleure, mais au prix d'un plus grand nombre de paquets émis.

Travaux en cours

De nouveaux travaux [4] étendent les recherches de ce chapitre. Leur objectif est de trouver des schémas de supervision explicites plus économes en énergie que les méthodes systématiques et cycliques présentes dans l'état de l'art. Partant du constat que la supervision active est toujours nécessaire et ne peut pas être facilement substituée, l'objectif consiste à trouver une méthode optimale de supervision permettant de réduire son coût autant que possible.

Mesure systématique

Une supervision cyclique impose à chaque nœud d'envoyer des rapports sur son état à intervalles réguliers. Le problème de cette approche est que son coût augmente linéairement avec le nombre de nœuds présents dans le LLN et peut être coûteux en ressources si la fréquence d'envoi est trop importante.

Une première piste de recherche en cours d'investigation consiste à diminuer le nombre d'informations envoyées à mesure que le comportement d'un nœud est stable. Cette méthode s'inspire de la méthode utilisée par Trickle [113] dans le protocole RPL qui diminue le nombre de messages de routage à mesure que le routage devient stable. L'évaluation de cette méthode notamment dans le cas de réseau instable et la comparaison avec des méthodes cycliques est l'un des premiers objectifs de ces nouveaux travaux.

Mesure ciblée dynamique

Une seconde approche consiste à choisir dynamiquement quel nœud interroger afin d'améliorer le modèle que la passerelle a d'un LLN. Cependant, la passerelle ne peut savoir si l'état d'un nœud diverge de son modèle avant de le lui demander. Ainsi la passerelle a le choix entre conserver un modèle incertain et ne pas solliciter un nœud donné afin d'économiser des ressources ou bien payer le coût d'une mesure active quitte à ce qu'elle ne soit pas pertinente. Cette problématique d'exploration contre exploitation [117] est connue et explorée dans le domaine de l'apprentissage renforcé [151].

Cette approche consiste à attribuer une valeur à l'information acquise quand la passerelle obtient une mesure venant d'un nœud et de mesurer ce gain au coût de son obtention. L'objectif est alors de maximiser la valeur des informations obtenues tout en minimisant le coût nécessaire pour l'obtenir.

Cependant, cette approche est complexe, car un nœud qui est instable temporairement peut présenter temporairement un gros gain d'information si la passerelle le choisit. Ainsi, il peut être tentant pour la passerelle de redemander plus régulièrement alors que le gain d'information était seulement temporaire.

Des approches venant du monde de la détection d'anomalies [117] et utilisant des Restless Multi-Armed Bandit (RMAB) sont une piste en cours d'exploration.

4. Entrepris lors d'un séjour au Swedish Institute of Computer Science (SICS) entre août et septembre 2015 en collaboration avec Simon Duquennoy.

Méthode d'évaluation

Le protocole d'évaluation de ces techniques serait d'évaluer l'information obtenue avec plusieurs méthodes (aléatoire, cyclique, dynamique, etc.) avec une approche de type oracle qui trouverait toujours la meilleure information à demander pour mettre à jour un modèle donné. Des évaluations sur des nœuds émulés et réels sont en cours de réalisation.

Conclusion

La supervision d'un LLN est importante pour garantir sa fiabilité et doit être réalisée en consommant aussi peu de ressources que possible. Ce chapitre montre comment une estimation implicite de l'utilisation de la radio peut être réalisée en utilisant la topologie réseau et le trafic observé au niveau de la passerelle pour inférer les temps d'utilisation de la radio de chaque nœud sans les solliciter.

La supervision passive peut fournir de manière transparente une estimation de l'utilisation de la radio qui peut ensuite être utilisée pour détecter des problèmes (retransmissions trop importantes, routeurs intermédiaires surchargés). Elle peut être utile dans le cas de nœuds contraints à l'extrême et qui ne peuvent pas fournir d'informations sur leur fonctionnement à une station de base ou bien pour implémenter des méthodes de supervision transparente pour les nœuds. Sa précision augmente à mesure que le réseau devient stable et le trafic prévisible. La connaissance de la topologie dans des scénarios maillés multi sauts est cruciale, car elle permet de prendre en considération les retransmissions effectuées par les nœuds routeurs.

Cependant, la supervision passive comporte plusieurs limitations, d'une part, les paquets émis localement ou perdus sont ignorés et induisent donc un biais qui tend à fournir des estimations sous-évaluées du temps d'utilisation de la radio. D'autre part, l'utilisation de cycle de veille asynchrone est certes efficace pour économiser de l'énergie, mais se paye par une absence d'informations sur le nombre de retransmissions qu'un émetteur va faire pour transmettre une trame. De plus, ce phénomène est amplifié dans des scénarios multi sauts où les erreurs d'estimations peuvent se faire sur chacun des sauts. Ainsi si la précision n'est pas jugée suffisamment bonne pour être utilisée en pratique il faut payer le coût de mesures explicites. Cependant, même dans ce cas, la supervision passive peut fournir des estimations de l'utilisation de la radio entre les mesures explicites.

La supervision passive peut s'appliquer à des couches MAC beaucoup plus régulières et fiables comme Time-Slotted Channel Hopping (TSCH) où les endormissements et les réveils sont déterministes (supprimant ainsi les phénomènes d'over-hearing, etc.). Ces méthodes d'accès étant plus prévisibles, elles peuvent être inférées plus facilement. Une ouverture possible de ces travaux serait de les adapter à ce type de couches MAC.

Des améliorations sont également possibles au niveau de l'inférence de l'impact des protocoles de routage lorsque la construction des routes est fiable (acquittement au moment de la construction des routes descendantes). Ainsi, une fois les routes établies, le routeur de bordure aurait la possibilité d'inférer le coût en transmission nécessaire pour l'établissement de cette topologie.

Aussi sophistiquée que soit l'inférence, la mesure active est indispensable pour un système ayant de fortes contraintes de fiabilité. La mesure passive n'a pas vocation à remplacer cette approche et se pose comme une source de métrique pouvant être déployée dans n'importe quelle condition et de manière transparente pour le LLN afin de fournir un service minimum de supervision.

Publications

Rémy Léone, Jérémie Leguay, Paolo Medagliani, Claude Chaudet. Tee : Traffic-based energy estimators for duty-cycled Wireless Sensor Networks. *IEEE International Conference on Communication*

(ICC), page 6749-6754, Londres, 2015.

Chapitre 4

Optimisation des ressources d'un LLN avec un cache intelligent

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

Sommaire

Ce chapitre présente comment optimiser l'utilisation des ressources d'un LLN par l'emploi d'un service de mise en cache qui adapte son comportement en fonction des conditions du LLN.

La section 4.1 introduit les mécanismes applicatifs mis en place au niveau de la passerelle et les contraintes liées à leur gestion.

La section 4.2 présente l'état de l'art sur les différentes solutions de mises en cache visant à optimiser l'utilisation des ressources d'un réseau.

La section 4.3 présente la contribution de ce chapitre en détaillant son architecture, ses modèles théoriques et les méthodes d'optimisation multi-objectifs employées.

La section 4.5 couvre la validation expérimentale du système proposé en présentant les gains obtenus.

La section 4.6 conclut ce chapitre et présente les ouvertures possibles de ces travaux.

Introduction

Les LLNs ont pour objectif de fournir des relevés et des mesures sur leur environnement à des services à valeur ajoutée. Comme vu dans la section 2.3, la passerelle a pour rôle d'assurer l'interface entre un LLN et ces services tiers.

Objectifs d'un Reverse Proxy Cache

Pour remplir son rôle d'interface applicative, la passerelle utilise un Reverse Proxy Cache (RPC) qui est un logiciel s'exécutant au niveau de la passerelle et qui intercepte l'ensemble des requêtes applicatives qu'elle reçoit. Un RPC a plusieurs objectifs : traduire les requêtes entre différents protocoles ; gérer les requêtes admises dans le LLN ; accélérer leur traitement tout en réduisant la consommation des ressources.

Traduction inter protocoles

Les nœuds capteurs mettent en jeu des protocoles applicatifs hétérogènes et supporter tous ces protocoles au niveau de chaque client serait coûteux et difficile à maintenir. Ainsi la passerelle utilise un mécanisme de *traduction* pour traduire les requêtes venant d'un protocole classique (par exemple HTTP) vers un protocole supporté par les nœuds du LLNs (par exemple Constrained Application Protocol).

Reverse-proxy

Les nœuds d'un LLN peuvent recevoir un flot trop important de requêtes applicatives qui aura un impact fort sur la consommation de leurs ressources. Afin de limiter ce type de situation, un reverse-proxy peut être utilisé afin de répondre à ces requêtes à la place des nœuds.

Un reverse-proxy est un logiciel qui se place entre un ou plusieurs serveurs et qui répond aux requêtes destinées à ces serveurs, à leur place en faisant les traitements nécessaires. Ce fonctionnement

est transparent pour les clients qui communiquent avec le reverse-proxy comme s'ils communiquaient avec les serveurs originaux.

Ce type d'architecture est utilisé par une grande partie des sites web les plus consultés du monde afin de router dynamiquement les requêtes vers différents serveurs afin de répartir la charge [158, 73].

Cache

Si un reverse-proxy n'a pas de cache, alors il se contente de faire suivre la requête vers le serveur visé. Dans le cas d'un LLN, le traitement d'une requête est long, car les latences entre les nœuds sont élevées (mécanisme de cycle de veille, réémissions sur le canal, etc.). D'autre part, lorsqu'une même requête est effectuée plusieurs fois sans que la réponse ne change, un LLN consomme de la bande passante et de l'énergie inutilement.

Ainsi, la passerelle utilise un mécanisme de cache afin d'économiser les ressources d'un LLN et accélérer le traitement des requêtes.

Fonctionnement d'un RPC

Le but d'un RPC est de réutiliser une réponse obtenue précédemment afin de répondre à une requête en cours en évitant de solliciter le LLN. Pour fonctionner, un RPC garde en mémoire une table de correspondance entre une URI et une réponse. Quand une nouvelle requête arrive, cette table est utilisée afin de la servir comme réponse sans solliciter de nouveau le serveur dont elle est issue.

Il existe deux mécanismes principaux : un mécanisme par fraîcheur ("Freshness model" en anglais) et un mécanisme par validation.

Fraîcheur des réponses

Une réponse à une requête qui est suffisamment "fraîche" peut être utilisée par le RPC pour répondre à de nouvelles requêtes. Un nœud capteur peut définir le temps maximal pendant lequel un RPC peut réutiliser une réponse en utilisant une option (Max-Age). Tant que la réponse n'est pas restée dans le RPC plus longtemps que cette durée, elle est considérée comme "fraîche" et le RPC l'utilise afin d'économiser les ressources du serveur d'origine. De plus, un nœud serveur qui ne veut pas que cette réponse soit mise en cache peut utiliser cette option en lui donnant une valeur nulle. Ce mécanisme est par exemple utile pour mettre en cache des mesures qui ne changent pas significativement au cours du temps (température, pluviométrie, etc.) [1].

Revalidation des réponses

Il peut arriver qu'une réponse soit toujours valide, mais ne soit plus considérée comme fraîche car son temps de validité a expiré. Dans ce cas, la réponse peut être revalidée par l'utilisation d'un mécanisme d'Entity Tag (ETag).

Un ETag est un identifiant unique assigné par le serveur applicatif à chaque version d'une ressource accessible par une URI. Si la ressource en question est modifiée, un nouveau ETag différent du précédent lui est assigné. Ainsi, un RPC peut comparer deux ETag successifs et vérifier s'ils correspondent à deux versions identiques d'une même ressource, et ainsi savoir si une demande peut être honorée par une réponse en cache ou pas.

Lorsqu'un client effectue une nouvelle requête, il peut indiquer quel était l'ETag de la précédente réponse obtenue. Dans le cas où elle est toujours valide, le RPC peut la revalider sans solliciter les nœuds. Dans le cas où le RPC a besoin de la revalider avec le serveur d'origine, il envoie l'ETag dans la requête. Si la réponse est toujours valide, un simple code de retour est renvoyé ce qui permet à

un nœud de revalider la réponse sans la transférer entièrement. Dans le cas contraire, la nouvelle réponse avec un nouveau ETag est transférée normalement.

Ce mécanisme est par exemple utile pour mettre en cache des notifications ou des alertes sur un état. Un état n'a pas de durée de vie connue à l'avance, car il peut changer n'importe quand, mais la réponse qu'il envoie sera revalidée en l'absence de changements.

Problématique des reverse proxy

L'utilisation d'un RPC requiert de le configurer correctement en raison de son rôle clé dans la gestion des requêtes applicatives. Un RPC est le logiciel en charge de la traduction des requêtes applicatives, ainsi s'il tombe en panne, cette traduction n'est plus assurée et les services tiers ne peuvent plus accéder aux LLNs par des requêtes usuelles. De plus, si un attaquant parvient à prendre le contrôle d'un RPC il peut effectuer des attaques Man-in-the-middle (MITM) qui peuvent à la fois compromettre les données, mais également permettre une attaque au déni de sommeil contre les nœuds afin d'épuiser leurs ressources.

D'autre part, un serveur applicatif contraint n'est pas toujours obligé de spécifier un temps de vie pour chaque ressource qu'il offre. Or si la quantité de requêtes pour ce nœud devient importante, l'absence de mécanisme de mise en cache dégradera les performances et consommera beaucoup de ressources. Ainsi quand un serveur applicatif ne le rejette pas explicitement, un RPC peut être légitimement mandaté pour spécifier des temps de vie dans le but d'alléger la charge sur un nœud applicatif. Cependant, le choix de la durée de vie pour une requête n'est pas trivial : un temps de vie trop court rendra la mémoire cache inefficace, car invalidée rapidement tandis qu'un temps de vie trop long risquera de donner des valeurs obsolètes. Dans un cas extrême, un reverse-proxy peut peupler son cache de manière permanente sans invalider aucune réponse et ainsi aucune requête applicative ne pourrait passer ce qui donnerait à la fois de grandes économies d'énergie et des réponses obsolètes rapidement. Ainsi un mécanisme visant à trouver des temps de validité optimaux pour chaque requête doit tenir compte de ces deux objectifs concurrents. Le reste de ce chapitre se concentrera sur les moyens d'obtenir ces temps de validité optimaux.

État de l'art

Les RPC peuvent être modélisés par des caches dits Time To Live où les réponses restent valides pendant un temps donné puis sont retirées du cache [21]. Ce cadre théorique mesure les performances attendues, mais ne donne cependant pas de critères sur comment choisir ces temps de vie en fonction d'un contexte donné.

RPC pour les LLNs

Les RPC sont utilisés dans les LLNs avec des architectures similaires à celles des sites web classiques [39, 40]. Cependant, le calcul des temps de vie n'est pas spécifié dans les implémentations et dépend des choix des utilisateurs.

L'utilisation de cache au niveau des routeurs intermédiaires du LLN est également une technique explorée [56], cependant les paramètres de gestion des temps de validité ne prennent généralement pas en compte les métriques d'un nœud (réserve de batterie) ou même du LLN entier (temps de vie).

Optimisation des ressources basée sur les temps de vie

Des méthodes heuristiques sont employées dans les RPC HTTP afin de configurer des temps de vie pour les réponses aux requêtes qui ne spécifient pas un temps de vie par elle-même [130]. Ces heuristiques sont très hétérogènes, mais se basent le plus souvent sur les dates de modification d'une URI pour fournir un temps de vie d'une réponse [70]. En outre, elles ne sont utilisées que lorsque le temps de vie n'est pas explicité par le serveur d'origine lui-même et ne peut pas dépasser une certaine durée sans l'envoi d'un avertissement à l'utilisateur final [1].

Ces approches ne sont pas satisfaisantes pour les LLNs, car le calcul des temps de vie ne se fait que sur l'URI et néglige le contexte propre à chaque nœud et en particulier, la quantité de paquets qu'il peut gérer et ses réserves d'énergies.

Optimisation des ressources énergétiques par cache

Les caches sont utilisés dans les Content Centric Network (CCN) pour distribuer le contenu dans le réseau aux meilleurs emplacements afin de répondre aux requêtes des utilisateurs efficacement. Les économies d'énergies sont des problématiques présentes au sein de cette communauté et des études ont montré comment utiliser les caches de manière optimale afin d'économiser de l'énergie.

Llorca [118] utilise des mécanismes distribués pour placer des caches de manière optimale dans un réseau afin d'économiser de l'énergie, or l'approche recherchée dans ce chapitre se concentre sur une architecture centralisée autour du RPC. D'autres approches [114, 33] utilisent un mécanisme de popularité d'un contenu afin de déterminer si une ressource doit être conservée ou non. Cependant, ces approches ne se concentrent que sur la présence d'un contenu dans un cache et ne spécifient pas explicitement un temps de vie pour cette ressource qui est l'objectif recherché.

Reverse Proxy Cache Adaptatif

Un Reverse Proxy Cache Adaptatif (RPCA) est un RPC qui configure pour chaque réponse de requête, un temps de vie optimal. Son objectif est de trouver en fonction des objectifs de durée de vie d'un LLN quelle est la qualité de service maximale qui peut être obtenue. Réciproquement, pour une qualité de service donnée, un RPCA donne les meilleurs objectifs de durée de vie possible.

Afin de trouver les temps de vie acceptables pour chaque ressource, un RPCA modélise la durée de vie d'un LLN et la satisfaction des utilisateurs comme deux fonctions des temps de vie des réponses qui doivent être optimisées en même temps. Puis le RPCA résout le problème multi-objectifs afin de fournir les temps de vie acceptables. La résolution d'un problème multi-objectifs étant coûteuse, il est important de trouver efficacement un ensemble de solutions afin de changer rapidement les temps de vie au cas où les hypothèses viendraient à changer.

Architecture

Un RPCA se place au niveau de la passerelle et intercepte comme le ferait un RPC classique les requêtes applicatives visant le LLN. Afin d'éviter les problèmes liés à la synchronisation des caches, il est supposé qu'il n'y a qu'un seul RPCA pour l'ensemble des nœuds. Cette hypothèse est assez réaliste, car dans les déploiements usuels, la passerelle est le point d'entrée unique vers un LLN.

Soit u, l'URI de la requête courante et c_u le temps de validité associé à la réponse de cette valeur. Une requête visant une URI u, venant de l'extérieur, en cache depuis moins de c_u sera traitée par le RPC sans solliciter le LLN. Soit $\mathscr{C}$ l'ensemble de tous les c_u que la passerelle gère.

1. Fixée à 24 heures dans le cas de HTTP [65].

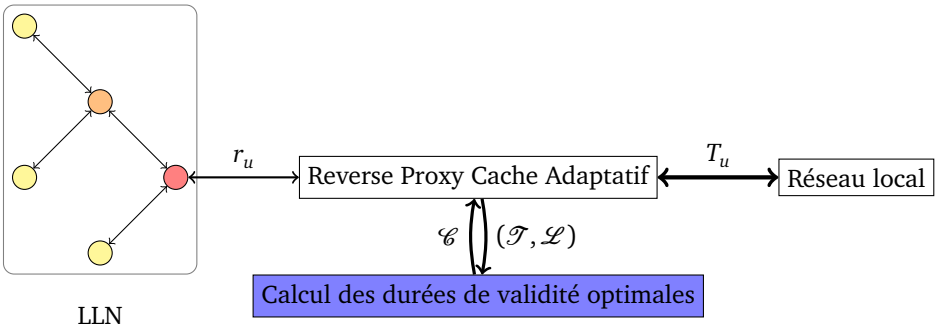


FIGURE 4.1 – Architecture du RPCA

Pour fonctionner, un RPCA a besoin d'estimer efficacement la durée de vie d'un LLN, or comme le montre le chapitre 3, la connaissance de la topologie est importante pour avoir des estimations correctes. Soit $\mathscr{T}$, la topologie du LLN et $\mathscr{L}$ la durée de vie souhaitée pour le LLN. Il est supposé dans le reste de ce chapitre que la topologie du réseau est connue du RPCA.

Calcul théorique de l'impact du cache sur l'intensité des requêtes

Le nombre moyen de requêtes reçues à destination de l'URI u est modélisé par un processus de Poisson de paramètre λ_u : $T_u = \frac{1}{\lambda_u}$ est le temps moyen entre deux requêtes successives vers u. Le RPCA agit sur les requêtes entrantes afin que le temps moyen entre deux requêtes pour la ressource u soit de r_u.

Si le RPCA a une valeur stockée qui est encore valide, il répond directement à la demande d'un client. Sinon, si la valeur requise n'est pas présente ou plus âgée que c_u, il transfère la demande au serveur du LLN hébergeant cette URI afin de le revalider ou à défaut d'avoir une nouvelle réponse.

Il est supposé que les temps caractéristiques de traitement d'une requête par le LLN sont très petits devant les temps c_u et T_u afin de supposer que quand une requête traverse le cache (miss), toutes les requêtes qui arrivent dans l'intervalle c_u qui suit seront servies par le cache (hit).

Soit m_u la probabilité qu'une requête pour l'URI u ne puisse pas être satisfaite par le RPCA.

À chaque requête qui traverse le cache (miss) est associée une fenêtre temporelle de taille c_u pendant laquelle toutes les requêtes sont traitées par le cache. Ces fenêtres ne se chevauchent pas [2] : les requêtes traversantes sont forcément espacées d'au moins c_u.

Si les requêtes venues de l'extérieur arrivent avec une intensité λ_u, le nombre de requêtes observées pendant un intervalle c_u est en moyenne $\lambda_u\, c_u$. Donc, à chaque requête qui traverse est associée (en moyenne) $\lambda_u\, c_u$ requêtes cache. La proportion de miss et de hit est donc :

$$m_u = \frac{1}{1+\lambda_u c_u} \tag{4.1}$$

$$h_u = 1 - m_u = \frac{\lambda_u c_u}{1+\lambda_u c_u} \tag{4.2}$$

Ainsi le LLN reçoit une intensité de $m_u\lambda_u$ requêtes. Ce processus n'est pas un processus de Poisson, en particulier, deux requêtes sont toujours espacées d'au moins c_u. Cependant, le calcul du temps moyen entre deux requêtes est toujours l'inverse de l'intensité. Ainsi, r_u se met sous la forme :

2. L'article original [111] fait une modélisation différente en supposant que le temps de vie est réinitialisé à chaque nouvelle requête.

$$r_u = \frac{1}{m_u \lambda_u} = T_u \left(1 + \frac{c_u}{T_u}\right) = T_u + C_u \tag{4.3}$$

Lorsque $c_u \to 0$, $r_u \to T_u$ ce qui correspond à un comportement où le cache est absent. D'autre part, quand c_u devient grand devant T_u, r_u tend vers c_u.

Modélisation de la durée de vie

Un RPCA a besoin d'avoir une estimation de la consommation énergétique afin d'établir une estimation de la durée de vie pour savoir si une configuration $\mathscr{C}$ est admissible.

Cette section détaille le modèle approché utilisé pour obtenir une estimation de la consommation énergétique. Comme pour le chapitre précédent, cette modélisation se concentre sur la consommation de l'interface radio qui est la principale source de consommation énergétique d'un nœud capteur [132] et néglige celle des autres composants (CPU, flash, etc.).

Il est supposé que les nœuds du LLN sont à la fois des serveurs applicatifs et des routeurs (en fonction de la topologie) et que leur batterie est une réserve fixe non renouvelable.

Énergie résiduelle

Pour un LLN utilisant une source d'énergie non rechargeable initiale de E_0, l'énergie résiduelle E_r à un instant T est donnée par :

$$E_r(T) = E_0 - \int_0^T P(t)\mathrm{d}t \tag{4.4}$$

où E_0 est l'énergie initiale du nœud considéré (en joules) et $P(t)$ est la puissance consommée (en watts) par le système au cours du temps. La durée de vie théorique du LLN est obtenue avec un modèle linéaire [32] qui peut être calculé en utilisant les r_u obtenus pour une configuration $\mathscr{C}$.

États de transmission d'un nœud

La radio d'un nœud peut être dans 4 états possibles : transmission, réception, en veille ou en écoute du canal. Ainsi, la consommation énergétique globale d'un nœud est obtenue en sommant les parts liées aux différents états :

$$\int_0^T P(t)\mathrm{d}t = \int_0^T P_{\mathscr{S}}\alpha_{\mathscr{S}}(t)\mathrm{d}t + \int_0^T P_{\mathscr{L}}\alpha_{\mathscr{L}}(t)\mathrm{d}t + \int_0^T P_{\mathrm{Tx}}\alpha_{\mathrm{Tx}}(t)\mathrm{d}t + \int_0^T P_{\mathrm{Rx}}\alpha_{\mathrm{Rx}}(t)\mathrm{d}t \tag{4.5}$$

$$= \alpha_{\mathscr{S}}P_{\mathscr{S}} + \alpha_{\mathscr{L}}P_{\mathscr{L}} + \alpha_{\mathrm{Tx}}P_{\mathrm{Tx}} + \alpha_{\mathrm{Rx}}P_{\mathrm{Rx}} \tag{4.6}$$

où : $\alpha_{\mathscr{S}}$ est la proportion de temps passé en sommeil ; $\alpha_{\mathscr{L}}$ est la proportion de temps passé en écoute du canal ; α_{Rx} est la proportion liée à la réception et α_{Tx} est la part liée à la transmission.

Dit autrement, $\Omega_X = \alpha_X P_X$ est la part de puissance liée à un état donné X où P_X est la puissance consommée dans l'état X qui est supposée constante au cours de la vie d'un nœud et α_X la proportion de temps passé en état X. La puissance consommée dans un état donné X étant constante, P_X peut être sortie de l'intégrale. De plus, puisque $\alpha_X(t)$ est une fonction créneau, son intégrale peut être simplifiée et donne simplement la proportion de temps passé dans l'état X.

Les sections suivantes se concentrent sur les moyens de modéliser et d'estimer les différents α_X

Modélisation des temps moyens de transmission et de réception avec Contiki-MAC

ContikiMAC est un mécanisme de cycle de veille disponible dans Contiki-OS [54] qui est notamment utilisé dans les nœuds capteurs pour économiser de l'énergie (cf. 3.3.3.2). Ce mécanisme de cycle de veille construit au-dessus de IEEE 802.15.4 permet au nœud d'éteindre et d'allumer sa radio périodiquement [3] pour consulter l'activité du canal.

Les réveils de nœuds voisins peuvent être désynchronisés, ainsi la source d'une trame devra l'envoyer à plusieurs reprises (*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 cependant ces optimisations n'offrent pas de garanties. Il est supposé ici que les paquets ne sont pas perdus en raison de collisions ou de congestions, car le trafic est jugé suffisamment faible.

T_C est défini comme la durée entre deux phases actives consécutives et $T_{\mathscr{S}}$ comme la durée de la phase de sommeil du nœud considéré. La durée de la phase active d'un nœud en l'absence de transmission peut être évaluée comme : $T_A = T_C - T_{\mathscr{S}}$.

Transmission d'un paquet

Les serveurs et routeurs transmettent des trames liées aux requêtes et aux réponses à travers le LLN. La durée de transmission d'un paquet de requête est définie comme : $T_r = \frac{L_r}{R}$ et celui de la réponse : $T_a = \frac{L_a}{R}$, 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 ContikiMAC [54] quand un nœud transmet un paquet p (une requête r ou une réponse a), il reste durant $T_{p,\mathrm{Tx}\to\mathrm{Tx}}$ en transmission et pour une période $T_{p,\mathrm{Tx}\to\mathrm{Rx}}$ en réception afin de recevoir le paquet Acknowledgement message (ACK) du destinataire. Ces périodes de temps peuvent changer en fonction de la synchronisation des réveils :

Cas favorable

$$T_{p,\mathrm{Tx}\to\mathrm{Tx}} = T_p \tag{4.7}$$

$$T_{p,\mathrm{Tx}\to\mathrm{Rx}} = T_{\mathrm{ACK}} \tag{4.8}$$

où T_p indique un paquet générique qu'il soit une requête T_r ou une réponse T_a. T_{ACK} est le temps nécessaire pour transmettre un ACK. Ce cas correspond à un nœud qui transmet quand le récepteur s'allume et ne transmet donc qu'une seule fois son paquet.

Cas défavorable

$$T_{p,\mathrm{Tx}\to\mathrm{Tx}} = T_p + \left\lceil \frac{T_{\mathscr{S}}}{T_p + T_d} \right\rceil T_p \tag{4.9}$$

$$T_{p,\mathrm{Tx}\to\mathrm{Rx}} = \left\lceil \frac{T_{\mathscr{S}}}{T_p + T_d} \right\rceil T_d + T_{\mathrm{ACK}} \tag{4.10}$$

3. Par défaut toutes les 125 ms

Le pire cas de transmission se produit quand le récepteur vient de s'endormir alors que le transmetteur veut commencer à envoyer sa trame. T_d est le temps que le transmetteur va passer en écoute juste après une tentative de transmission pour détecter son éventuel succès par un acquittement d'un récepteur juste après l'envoi d'une trame de donnée. Le transmetteur va envoyer la trame tous les $T_p + T_d$ jusqu'à ce que le récepteur soit éveillé au début d'une transmission. À chaque tentative, le transmetteur va écouter le canal pendant T_d pour détecter un éventuel acquittement. Ainsi il va effectuer $\lceil \frac{T_{\mathscr{I}}}{T_p+T_d} \rceil$ transmissions sans succès puis quand le récepteur se réveille, il devra transmettre encore une fois pour que la trame soit finalement reçue et acquittée.

Cas moyen

Les cycles d'endormissement et de réveils sont désynchronisés. Ainsi le récepteur peut se réveiller à n'importe quel moment : tous les cas sont équiprobables. Donc, le cas moyen est obtenu en faisant la moyenne entre le meilleur et le pire cas :

$$T_{p,\mathrm{Tx}\to\mathrm{Tx}} = \frac{T_p}{2}\left\lceil \frac{T_{\mathscr{I}}}{T_p + T_d} \right\rceil + T_p \tag{4.11}$$

$$T_{p,\mathrm{Tx}\to\mathrm{Rx}} = \frac{T_d}{2}\left\lceil \frac{T_{\mathscr{I}}}{T_p + T_d} \right\rceil + T_{\mathrm{ACK}} \tag{4.12}$$

L'équation (4.11) 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).

Réception d'un paquet

Quand un nœud reçoit un paquet, il passe une partie de son temps en réception ($T_{\mathrm{Rx}\to\mathrm{Rx}}$) et une partie de son temps en transmission ($T_{\mathrm{Rx}\to\mathrm{Tx}}$) puisqu'il a besoin de transmettre un ACK à l'expéditeur du paquet. Ces périodes de temps peuvent être exprimées par :

Cas favorable

$$T_{p,\mathrm{Rx}\to\mathrm{Rx}} = T_p \tag{4.13}$$

$$T_{p,\mathrm{Rx}\to\mathrm{Tx}} = T_{\mathrm{ACK}} \tag{4.14}$$

Le récepteur se réveille juste avant le début de l'émission et ne l'écoute donc qu'une seule fois $T_{p,\mathrm{Rx}\to\mathrm{Rx}} = T_p$.

Cas défavorable

$$T_{p,\mathrm{Rx}\to\mathrm{Rx}} = 2T_p \tag{4.15}$$

$$T_{p,\mathrm{Rx}\to\mathrm{Tx}} = T_{\mathrm{ACK}} \tag{4.16}$$

Le récepteur se réveille juste après le début de la transmission ainsi le nœud rate la première et doit en avoir une seconde pour recevoir correctement la trame. Ainsi dans le pire cas $T_{p,\mathrm{Rx}\to\mathrm{Rx}} = 2T_p$.

Cas moyen

En utilisant le même argument d'équiprobabilité que précédemment, le cas moyen est obtenu en faisant la moyenne entre le meilleur et le pire cas :

$$T_{p,\mathrm{Rx}\to\mathrm{Rx}} = \frac{3T_p}{2} \tag{4.17}$$

$$T_{p,\mathrm{Rx}\to\mathrm{Tx}} = T_{\mathrm{ACK}} \tag{4.18}$$

où T_p comme pour le cas précédent, est le temps nécessaire pour transmettre un paquet et T_{ACK} est la durée de transmission d'une trame d'acquittement ACK. L'équation (4.17) 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 et où une seconde écoute est nécessaire.

Consommation des serveurs applicatifs

Les termes α_{Tx} et α_{Rx} vont être différents selon que le nœud est un routeur ($\alpha_{router,\mathrm{Tx}}$, $\alpha_{router,\mathrm{Rx}}$) ou un serveur applicatif ($\alpha_{server,\mathrm{Tx}}$, $\alpha_{server,\mathrm{Rx}}$). Soit r_i le temps moyen entre deux requêtes consécutives pour un nœud i. Soit T_r : le temps nécessaire pour transmettre une requête applicative et T_a le temps nécessaire pour transmettre la réponse a cette requête. P_{Tx} et P_{Rx} désignent respectivement la puissance consommée par un nœud en phase de transmission et de réception.

L'objectif consiste à modéliser l'impact énergétique de la transmission de ces requêtes applicatives d'une manière similaire à celle utilisée dans le chapitre 3.3 :

$$\alpha_{server,\mathrm{Tx}} = \frac{T_{r,\mathrm{Rx}\to\mathrm{Tx}} + T_{a,\mathrm{Tx}\to\mathrm{Tx}}}{r_i} \tag{4.19}$$

$$\alpha_{server,\mathrm{Rx}} = \frac{T_{r,\mathrm{Rx}\to\mathrm{Rx}} + T_{a,\mathrm{Tx}\to\mathrm{Rx}}}{r_i} \tag{4.20}$$

Consommation des nœuds routeurs

Les routeurs servent d'intermédiaires aux transmissions des nœuds pour lesquels ils offrent des routes.

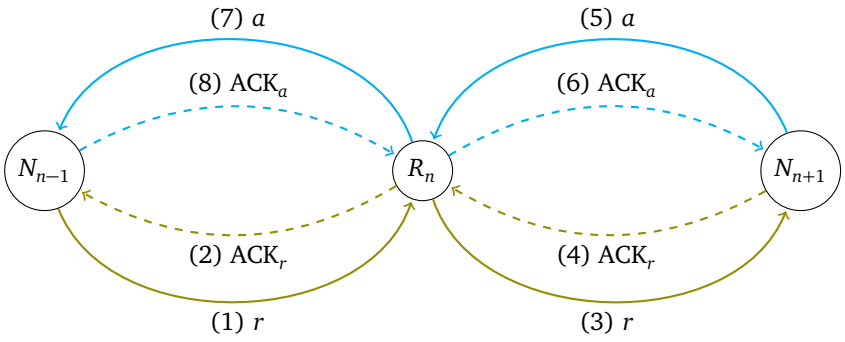


FIGURE 4.2 – Schéma de transmission d'un paquet par routeur

La Figure 4.2 illustre les envois de messages survenant à un routeur R_n routant une requête applicative et sa réponse d'un nœud N_{n-1} vers un nœud N_{n+1}. Le temps passé en transmission et en réception pour le routeur R_n lors du traitement complet de cette requête s'obtient de la façon suivante :

$$T_{router,\mathrm{Tx}} = T_{r,\mathrm{Tx}\to\mathrm{Tx}} + T_{r,\mathrm{Rx}\to\mathrm{Tx}} + T_{a,\mathrm{Tx}\to\mathrm{Tx}} + T_{a,\mathrm{Rx}\to\mathrm{Tx}} \tag{4.21}$$

$$T_{router,\mathrm{Rx}} = T_{r,\mathrm{Rx}\to\mathrm{Rx}} + T_{r,\mathrm{Tx}\to\mathrm{Rx}} + T_{a,\mathrm{Rx}\to\mathrm{Rx}} + T_{a,\mathrm{Tx}\to\mathrm{Rx}} \tag{4.22}$$

Le temps d'utilisation de la radio d'un routeur est modélisé par la somme des temps de traitement des paquets routés, car il gère à la fois l'acheminement de la requête et de sa réponse :

$$\alpha_{router,\mathrm{Tx}} = \sum_{j\in m_i} \frac{T_{router,\mathrm{Tx},j}}{r_j} \tag{4.23}$$

$$\alpha_{router,\mathrm{Rx}} = \sum_{j\in m_i} \frac{T_{router,\mathrm{Rx},j}}{r_j} \tag{4.24}$$

où m_i désigne l'ensemble des nœuds pour lesquels le routeur i route des paquets. Si un routeur agit également comme serveur applicatif alors il faut également rajouter les termes issus de l'équation 4.19 et 4.20. Dans le cas particulier de la passerelle, les α s'obtiennent en observant que la passerelle est routeur pour tous les nœuds du LLN.

Consommation en phase d'écoute de canal

Les phases d'écoute du canal se produisent à chaque cycle qu'une trame de donnée soit présente ou non. Il est supposé qu'en l'absence de transmission des voisins, la part d'écoute est $\frac{T_A}{T_C}$ et que le rapport est préservé dans les cas de transmissions (hors périodes de transmission et de réception). La consommation de cette phase d'écoute peut donc s'exprimer comme :

$$\alpha_{\mathscr{L}} = \frac{T_A}{T_C}(1 - \alpha_{\mathrm{Tx}} - \alpha_{\mathrm{Rx}}) \tag{4.25}$$

Consommation en phase de sommeil

La phase de sommeil s'obtient en retirant le temps passé dans les 3 autres états : en écoute, en transmission et en réception.

$$\alpha_{\mathscr{S}} = \frac{T_C - T_A}{T_C}(1 - \alpha_{\mathrm{Tx}} - \alpha_{\mathrm{Rx}}) \tag{4.26}$$

Optimisation multi-objectifs

La modélisation de la durée de vie permet de lier la quantité de requêtes qu'un nœud doit gérer à l'énergie qu'il va dépenser pour le faire. Cependant, dans ces conditions un reverse-proxy cherchant à optimiser la durée de vie rendrait cette quantité de requêtes aussi faible que possible en augmentant sans limites les durées de temps de vie en cache afin d'économiser de l'énergie. Cependant, un utilisateur veut pouvoir disposer de réponses suffisamment fraîches pour être satisfait. Ainsi le problème

d'optimisation des ressources peut être modélisé comme une optimisation multi-objectifs cherchant à maximiser à la fois la durée de vie et la satisfaction des utilisateurs tout en respectant un ensemble de contraintes.

Ce problème admet plusieurs solutions qui font chacune un compromis différent entre consommation et satisfaction. Il existe deux façons de faire ce choix : a priori ou a posteriori.

Raisonner *a priori* signifie que le RPCA va définir précisément quelle solution est souhaitable en utilisant une fonction d'*utilité* définie pour chaque solution possible. L'avantage de cette méthode est qu'elle permet de ramener un problème multi-objectifs en un problème à un seul objectif plus facile à résoudre. Cependant, fonctionner a priori implique d'exécuter l'optimisation à chaque fois que les préférences de l'utilisateur changent, ce qui est coûteux. D'autre part, définir cette fonction d'utilité est complexe, car elle doit prendre en compte de multiples préférences et introduire des priorités dans les différentes fonctions objectives.

Une alternative consiste à fonctionner a posteriori dans laquelle autant de solutions optimales que possible sont produites dans un premier temps. Puis parmi toutes les solutions admissibles proposées, l'administrateur en choisit une en fonction de ses choix et n'a ainsi pas à écrire de fonction d'utilité et se contente de choisir parmi un ensemble restreint de solutions. Cette approche en plus d'être plus simple à gérer pour l'administrateur a un autre avantage : les solutions optimales peuvent être gardées en mémoire. Ainsi si l'administrateur souhaite modifier sa solution, il n'est pas nécessaire de refaire tourner une seconde fois l'optimisation multi-objectifs.

Paramètres du problème

Solution

Un problème d'optimisation est dit multi-objectifs lorsque plusieurs objectifs doivent être maximisés en même temps. Le RPCA utilise une modélisation dans le but d'obtenir un ensemble de solutions optimales pour chacune de ses fonctions objectives. Ces solutions forment ce qu'on appelle un front non dominé de Pareto qui est l'ensemble des solutions pour lesquelles il n'est pas possible de maximiser une des fonctions objectives sans en réduire une autre.

Une solution $\mathscr{C}$ est composée de l'ensemble des temps de vie des réponses c_u pour chaque URI u. Quand le RPCA utilise une solution $\mathscr{C}$, la quantité de requêtes admises dans le LLN pour le nœud i sera de r_i.

Contraintes pour les durées de validité des réponses

Un c_u est compris entre deux bornes : $c_{max}(u)$ qui est la durée maximale qu'une réponse u peut rester en cache et $c_{min}(u)$ qui est la durée minimale.

$$c_{min}(u) \leq c_u \leq c_{max}(u)$$

Dans le cas où le seul objectif est la durée de vie du réseau (respectivement la satisfaction des utilisateurs), toutes les durées de temps de validité des réponses doivent être mises au maximum (respectivement minimum) admissible : $\mathscr{C} = \mathscr{C}_{max} = c_{max}(u), \forall u$ (respectivement $\mathscr{C} = \mathscr{C}_{min} = c_{min}(u), \forall u$.

Satisfaction d'un utilisateur

La modélisation de la satisfaction d'un utilisateur est basée sur le temps de validité d'une ressource : plus une information est récente plus l'utilisateur est satisfait. Ainsi, soit γ_u mesurant la

satisfaction d'un utilisateur par une variable normalisée et linéaire [77] ("min-max scaling" en anglais) :

$$\forall u, \gamma_u = \frac{c_{max}(u) - c_u}{c_{max}(u) - c_{min}(u)} \tag{4.27}$$

En particulier, une satisfaction est minimale (égale à 0) quand $c_u = c_{max}$ et une satisfaction est maximale (égale à 1) quand $c_u = c_{min}$.

Résolution du problème multi-objectifs

Résoudre un problème multi-objectifs n'est pas trivial, car il existe le plus souvent plusieurs solutions qui sont optimales au sens de Pareto. D'autre part, résoudre le problème multi-objectifs dans le cas du RPCA est difficile en raison de l'espace des solutions possibles, des fonctions objectifs et des contraintes sur les ressources disponibles pour effectuer cette optimisation.

Grand espace de recherche

Un RPCA gère un grand nombre d'URI, ainsi chercher sur l'espace complet des solutions est difficilement envisageable. D'autre part, une solution approchée peut être acceptable si elle consomme une quantité limitée de ressources supplémentaires par rapport à une solution optimale. Il n'est pas pertinent de dépenser de grandes quantités de calculs pour trouver les temps de cache optimaux si des méthodes approchées sont disponibles et moins coûteuses.

Fonctions objectives quelconques

Les LLNs sont des systèmes hétérogènes et les fonctions qui modélisent les objectifs (durées de vie, satisfaction des utilisateurs, etc.) peuvent être complexes. Ainsi, trouver les c_u adaptés pour chaque ressource ne peut reposer sur des hypothèses à propos des fonctions qui modélisent le LLN ; elles sont quelconques.

Économie de temps de calcul et de mémoire

Résoudre un problème d'optimisation est coûteux en temps de calcul et en mémoire quand le nombre de variables est grand ou qu'il devient complexe de calculer une fonction objectif. D'autre part, en cas de changement des hypothèses du problème (pic de trafic, changements d'un seuil de satisfaction, etc.), le RPCA doit pouvoir changer rapidement sa configuration afin de s'adapter aux nouvelles hypothèses. Ainsi il faut pouvoir conserver plusieurs $\mathscr{C}$ afin de changer les temps de vie sans refaire les calculs depuis le début.

État de l'art

Les méta-heuristiques sont des algorithmes généralistes d'optimisation visant à offrir une solution approchée à une large gamme de problèmes différents en utilisant les mêmes méthodes génériques d'un problème à l'autre. Le champ des méta-heuristiques est vaste [173, 121] ainsi il est nécessaire de sélectionner les caractéristiques recherchées :

Un RPCA doit pouvoir avoir plusieurs $\mathscr{C}$ afin de trouver des alternatives rapidement en cas de changement des pics de trafic. Un algorithme ne produisant qu'une solution $\mathscr{C}$ devrait être relancé à chaque fois qu'un changement des consignes de l'administrateur arrive. Ainsi, il est préférable d'utiliser des méthodes produisant des populations de solutions qui produisent plusieurs solutions proches

du front de Pareto afin qu'en cas de changement, un administrateur puisse disposer d'emblée de solutions alternatives.

D'autre part, il est souhaitable d'avoir des solutions hétérogènes correspondant à un vaste éventail de configurations possibles qui peuvent être changées en fonction des conditions du réseau (changement du trafic entrant, etc.). Ainsi des méthodes de recherche globales sont plus adaptées pour trouver des solutions très diverses à l'inverse d'une recherche locale qui va privilégier l'amélioration de solutions existantes autour de certaines solutions.

De nombreuses approches par populations sont possibles [179], parmi les plus connues on peut citer les algorithmes génétiques qui sont connus et utilisés dans les LLNs pour avoir un placement optimal des nœuds [64, 90], pour construire des clusters efficaces en énergies [88, 4] ou encore effectuer des partitions d'un LLN efficaces en énergie [105]. Les algorithmes génétiques sont particulièrement adaptés pour trouver des approximations à des problèmes en ne faisant aucune hypothèse sur l'espace des solutions possibles ou sur les fonctions objectifs qui sont considérées comme des boites noires.

Ainsi, un algorithme génétique est une méthode générique pour obtenir une approximation de la solution à un problème complexe et sera donc appliqué ici à la résolution du problème d'optimisation multi-objectifs.

Formalisation en algorithme génétique

Un algorithme génétique permet d'approcher une solution à un problème d'optimisation en utilisant une heuristique proche de celle de l'évolution [121].

Une solution au problème multi-objectifs y est représentée sous la forme d'un individu composé de différents gènes. Dans le cas du RPCA, un individu est une solution $\mathscr{C}$ contenant un ensemble de temps de vie c_u pour chaque URI u. Un algorithme génétique se termine soit quand le nombre maximal d'itérations (appelée génération) est atteint, soit quand la population a atteint un niveau de qualité acceptable, soit quand le budget de calcul a été atteint. Ainsi un RPCA peut être configuré pour fournir une solution en fonction d'un budget prédéfini de calcul.

D'autre part, ces algorithmes ne font pas d'hypothèse sur les fonctions objectifs qui sont considérées comme des boites noires. Ainsi, les fonctions objectifs modélisant le problème peuvent être quelconques.

Fonctionnement global

La Figure 4.3 illustre le mécanisme de fonctionnement d'un algorithme génétique.

Un algorithme génétique démarre en utilisant une population d'individus de base qui est représentée dans l'étape 1 sur la Figure 4.3. Dans le cas du RPCA, la population initiale est générée en tirant un c_u au hasard entre c_{min} et c_{max} afin d'éviter les solutions inadmissibles d'emblée.

Cette population va subir un traitement systématique pendant un nombre fini d'itérations appelées *générations*. Ce traitement est composé de plusieurs étapes : évaluation des aptitudes de chaque solution, sélection des meilleurs individus, croisement et application des mutations. Ces étapes permettent de créer une nouvelle population et ce cycle recommence pendant un nombre fixé de générations ou bien lorsque les populations sont acceptables. Les meilleures solutions qui ont été sélectionnées au fil des générations sont données comme solution au problème et constituent une approximation du front de Pareto.

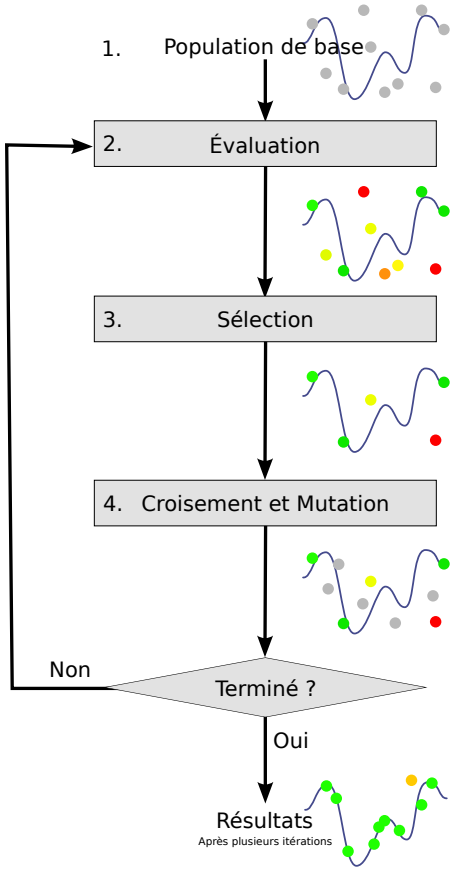


FIGURE 4.3 – Schéma des étapes d'un algorithme génétique

Aptitude

La fonction d'aptitude (ou "fitness" en anglais) évalue à quel point une solution est admissible ou non et si elle l'est quelle est son efficacité. L'algorithme génétique n'impose aucune hypothèse sur cette fonction qui doit seulement renvoyer un réel.

Déterminer une fonction d'aptitude est la partie la plus liée au problème multi-objectifs lors de la conception d'un algorithme génétique ; les autres étapes étant relativement génériques d'un problème à l'autre.

Pour chaque $\mathscr{C}$, il y a deux critères à prendre en compte : la durée de vie minimale du LLN qu'il permet d'obtenir et la satisfaction que les utilisateurs auront si ce $\mathscr{C}$ est utilisé. Soit $\mathscr{L}(\mathscr{C})$ la durée de vie obtenue dans le pire cas c'est-à-dire quand $r_u = c_u$. Soit $\Gamma(\mathscr{C})$ qui est la satisfaction moyenne ressentie par l'utilisation et qui est définie comme : $\Gamma(\mathscr{C}) = \frac{1}{N}\sum_u \gamma_u$.

Soit f la fonction d'aptitude obtenue en normalisant les grandeurs $\Gamma(\mathscr{C})$ et $\mathscr{L}(\mathscr{C})$ entre 0 et 1 :

$$f(\mathscr{C}) = \frac{1}{2}\left(\frac{\Gamma(\mathscr{C})-\Gamma(\mathscr{C}_{\min})}{\Gamma(\mathscr{C}_{\max})-\Gamma(\mathscr{C}_{\min})} + \frac{\mathscr{L}(\mathscr{C})-\mathscr{L}(\mathscr{C}_{\min})}{\mathscr{L}(\mathscr{C}_{\max})-\mathscr{L}(\mathscr{C}_{\min})}\right) \tag{4.28}$$

$$= \frac{1}{2}\left(\Gamma(\mathscr{C}) + \frac{\mathscr{L}(\mathscr{C})-\mathscr{L}(\mathscr{C}_{\min})}{\mathscr{L}(\mathscr{C}_{\max})-\mathscr{L}(\mathscr{C}_{\min})}\right) \tag{4.29}$$

Or, puisque $\Gamma(\mathscr{C}_{\min}) = 0$ et $\Gamma(\mathscr{C}_{\max}) = 1$, l'équation (4.28) peut être simplifiée pour obtenir (4.29). D'autre part, la moyenne permet de ne pas privilégier des solutions conservatrices en énergie au détriment d'autres.

Sélection

Le processus de sélection utilise l'aptitude précédemment définie pour sélectionner les solutions qui vont se "reproduire". Un des écueils courant des méthodes d'optimisation multi-objectifs génétique est de rester bloqué dans un sous-ensemble de solutions très homogènes. Dans le cas du RPCA cela voudrait dire que les solutions resteraient fixées sur un point localement optimal, or l'objectif est d'avoir une population aussi diversifiée que possible afin de pouvoir changer rapidement de solution en cas de changement de trafic.

Ainsi, à niveau d'aptitudes semblables, les solutions les plus éloignées les unes des autres devraient être sélectionnées. C'est la méthode utilisée par NSGA-II [47] qui est élitiste : les meilleures solutions sont retenues d'une génération à l'autre et forment les élites. Cependant, cette sélection s'accompagne d'une préservation de la diversité qui privilégie les solutions éloignées les unes des autres.

Pour fonctionner, NSGA-II utilise une métrique d'estimation de densité qui va estimer la distance moyenne qui le sépare des autres solutions. Puis lorsque deux solutions doivent être départagées lors d'une sélection, celle qui est la plus éloignée des autres solutions sera choisie. Cette méthode de sélection permet d'avoir un large ensemble de solutions et sera celle utilisée pour le reste du chapitre.

D'autre part, cette méthode garde en mémoire les meilleures solutions vues à chaque génération ("hall of fame" en anglais) afin de ne perdre aucune solution intéressante même si par le hasard des sélections elle n'est pas sélectionnée.

Croisement & Mutation

Les algorithmes génétiques utilisent des opérateurs afin d'altérer les populations de solutions existantes pour en former de nouvelles et explorer ainsi l'espace des solutions. Il existe essentiellement deux opérateurs génétiques : le croisement et la mutation.

Comme de nombreuses méta-heuristiques, ces mécanismes très hétérogènes peuvent être complexes à configurer, car de nombreux paramètres sont disponibles. Cependant, l'utilisation d'algorithmes de sélection élitiste permet de mitiger cette complexité, car les meilleures solutions sont gardées en mémoire dans le "hall of fame". Ainsi, les mécanismes de mutation et de croisement peuvent être utilisés ensemble et configurés pour être "exploratoires", tandis que les mécanismes de sélection s'occupent de garder les meilleures solutions.

Mutation

La mutation est un processus de transformation qui est appliquée sur chaque individu avec une probabilité p_m à chaque génération. Ce processus modifie les gènes d'une solution (en l'occurrence les c_u) dans les bornes inférieures et supérieures admissibles (en l'occurrence c_{min} et c_{max}).

Le reste de cette thèse utilise la méthode préconisée par les auteurs de NSGA-II qui est une mutation polynomiale qui utilise une distribution polynomiale définie entre c_{min} et c_{max} [48].

Croisement

Le croisement est un mécanisme d'exploration qui prend deux sous-ensembles de c_u, c'_u appartenant à deux solutions distinctes $\mathscr{C}$ et $\mathscr{C}'$ et les échangent à la génération suivante.

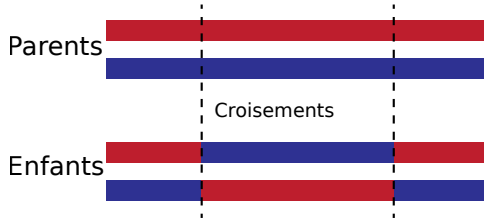


FIGURE 4.4 – Croisement de deux individus par points doubles

La Figure 4.4 illustre ce mécanisme de croisement à point double qui sélectionne deux points et échange l'ensemble des gènes entre ces deux points. C'est le mécanisme de croisement qui est recommandé pour être appliqué à NSGA-II [121] et qui sera utilisé dans cette thèse.

Validation expérimentale

Le RPCA est évalué dans le scénario suivant : 12 serveurs CoAP utilisant la bibliothèque Erbium [101] sont déployés. Ces nœuds utilisent Contiki [53] comme système d'exploitation et sont émulés via Cooja [138]. Cooja les émule comme des TmoteSky fonctionnant avec un processeur de type MSP430 et utilisant pour communiquer une puce radio CC2420 dont les consommations énergétiques sont connues [149].

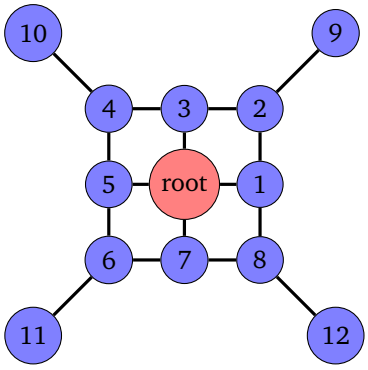


FIGURE 4.5 – Topologie radio considérée pour la validation expérimentale du RPCA.

La Figure 4.5 représente la topologie radio utilisée où un lien signifie que les nœuds peuvent être en communication radio l'un avec l'autre. Les nœuds utilisent le protocole RPL [197] pour établir une topologie de routage qui est connue par le routeur de bordure afin de modéliser la durée de vie du réseau. La passerelle se trouve au centre de cette topologie : elle est la racine de cette topologie de routage.

Le tableau 4.1 donne les paramètres numériques utilisés lors de l'expérience. Les nœuds utilisent ContikiMAC comme mécanisme de cycle de veille afin d'économiser de l'énergie et démarrent tous avec la même énergie initiale. La consommation énergétique du réseau est obtenue en utilisant la topologie et la modélisation de ContikiMAC précédemment introduite. La durée de vie du LLN est calculée comme l'intervalle de temps entre son démarrage et la perte de son premier nœud.

Le RPCA met en cache les réponses rendues par les nœuds afin de les mettre à disposition pour une éventuelle autre demande entrante. Une fois la topologie et le trafic connus, la résolution du problème

Débit de transmission	R	250 kbps
Intervalle entre deux tentatives de préambules	T_p	0.4 ms
Temps d'une période MAC	T_C	0.244 s
Temps d'une période active	T_A	40 ms
Temps pour détecter un paquet ACK	T_d	0.16 ms
Taille d'une requête CoAP (`GET`)	L_r	87 octets
Taille d'une réponse CoAP	L_a	96 octets
Temps pour transmettre un IEEE 802.15.4 ACK	T_{ACK}	0.608 ms
Puissance consommée lors de la transmission	P_{Tx}	0.0511 W
Puissance consommée lors de la réception	P_{Rx}	0.0588 W
Puissance consommée lors du sommeil	$P_{\mathscr{S}}$	$2.4 \cdot 10^{-7}$ W
Nombre de nœuds dans le LLN	N	12
Nombre de run par configuration		10
Nombre de requêtes CoAP traité par nœuds		50

TABLE 4.1 – Paramètres utilisés dans les simulations.

multi-objectifs est lancée sur la passerelle et les différents $\mathscr{C}$ sont calculés. Le RPCA a suffisamment de mémoire pour retenir toutes les réponses pour chaque URI disponible dans le réseau. La passerelle est toujours allumée, car non contrainte en énergie et assure une connexion permanente à un réseau local conventionnel.

Des requêtes HTTP sont envoyées vers le RPCA puis sont traduites vers une requête CoAP et inversement en utilisant la bibliothèque Californium [102]. Le trafic ne démarre que lorsque la topologie de routage est établie.

Le temps entre chaque requête est modélisé par un temps d'attente exponentiel de paramètre λ_u et $\forall u, \lambda_u = \lambda$. Ainsi le temps entre chaque requête correspond à r_u pour le nœud qui héberge la ressource u. 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 ne causant pas de fragmentations. Il est supposé que le trafic est suffisamment faible pour que les pertes de paquets soient négligeables.

Compromis entre durée de vie et satisfaction utilisateur

Le RPCA est configuré pour utiliser un algorithme génétique pour trouver les solutions avec les paramètres suivants :

Les probabilités d'accouplement et de mutation sont laissées aux valeurs recommandées par les créateurs de NSGA-II. Le nombre de générations et la taille de population sont choisis afin d'avoir un temps de calcul modeste et d'obtenir une approximation du front de Pareto rapide. La Figure 4.6 montre la population de solutions produites par l'algorithme génétique. Chaque point solution est un $\mathscr{C}$ contenant l'ensemble des c_u. Ces points forment une approximation du front de Pareto qui représente les solutions optimales et les durées de vie mesurées en jours.

Les poins $\mathscr{C}_{min}$ et $\mathscr{C}_{max}$ sont également présentés afin de montrer les bornes possibles des solutions. Ainsi $\mathscr{C}_{min}$ garanti une fraîcheur aussi élevée que possible, mais raccourcit la durée de vie alors que $\mathscr{C}_{max}$ garanti une durée de vie aussi longue que possible.

Une valeur haute de c_u sera utilisée pour des nœuds ayant peu d'énergie. Par contre, une valeur

Population	P	100
Probabilité de croisement	p_c	0.5
Probabilité de mutation	p_m	0.2
Nombre de générations	n_g	50
Temps de validité en cache minimal	c_{min}	1s
Temps de validité en cache maximal	c_{max}	9s

TABLE 4.2 – Paramètres utilisés dans l'optimisation multi-objectifs.

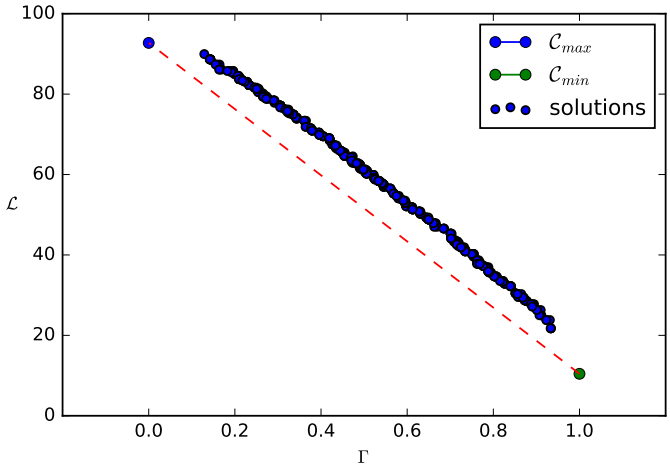


FIGURE 4.6 – Front de Pareto pour le scénario envisagé.

c_u petite sera pour des informations devant être aussi récentes que possible.

Cache hit

La Figure 4.7 illustre l'évolution du nombre de requêtes servies par le cache (hit) en fonction de la durée de vie des requêtes c_u choisies. La simulation (traits pleins) est comparée à la modélisation théorique (lignes en pointillés cf (4.2)) pour différentes valeurs de paramètres λ sur la Figure 4.7. Afin de simplifier la représentation visuelle, le graphique montre quels sont les résultats obtenus lorsque tous les c_u et λ_u sont égaux entre eux.

Plus c_u est grand par rapport à $T_u = \frac{1}{\lambda_u}$, plus le cache est efficace, car la probabilité que la requête soit servie par le RPCA augmente épargnant ainsi aux nœuds du LLN de les traiter.

Quand c_u est petit devant $T_u = \frac{1}{\lambda_u}$, la fréquence de requête est plus faible que le temps de vie dans le cache. Le cache est ici inefficace, car une réponse n'y vit pas assez longtemps pour être susceptible d'être utilisée.

Ainsi l'efficacité d'un cache n'est pas seulement liée au temps de vie d'une requête, mais également à son adéquation avec le trafic qu'il reçoit.

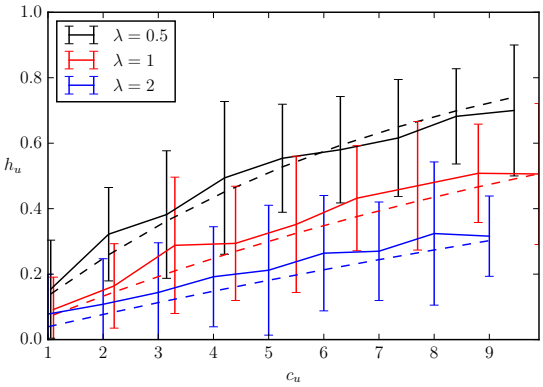


FIGURE 4.7 – Évolution du cache hit en fonction de la durée de vie

Amélioration de la durée de vie

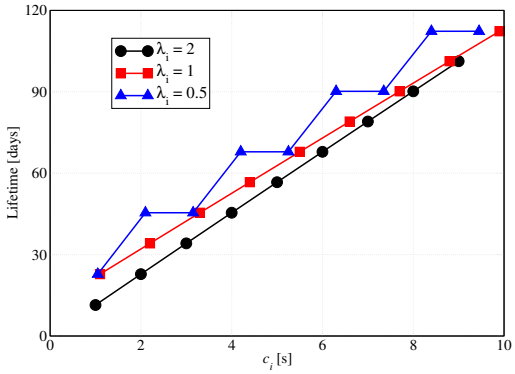


FIGURE 4.8 – Évolution de la durée de vie théorique en fonction des c_u

La Figure 4.8 illustre la durée de vie théorique obtenue pour le LLN pour différentes configurations de cache identiques. Il est observé que plus les temps de cache sont élevés, plus la durée de vie du LLN est élevée ce qui est attendu, car une grande durée de vie implique que moins de requêtes applicatives sont gérées par le LLN.

Conclusion

Ce chapitre montre comment un RPCA permet d'optimiser l'utilisation des ressources d'un LLN en adaptant la quantité de requêtes applicatives admises dans le LLN. Ce mécanisme utilise une modélisation de la consommation énergétique et de la satisfaction des utilisateurs afin de fournir un temps de vie pour toutes les réponses aux requêtes que le RPCA doit gérer.

Un RPCA change dynamiquement la quantité de requêtes applicatives entrantes dans un LLN afin d'économiser ses ressources d'une part et d'accélérer le traitement des requêtes d'autre part. De plus, le RPCA utilise une estimation de la durée de vie pour aider l'administrateur d'un LLN à trouver une configuration adaptée pour les objectifs de durée de vie visés.

L'utilisation d'algorithme génétique permet d'avoir un mécanisme général pouvant être appliqué à n'importe quelle situation. Cependant, cette méthode n'offre pas de garanties de convergence vers le front de Pareto pour un nombre de générations donné. Des méthodes plus spécifiques utilisant notamment une hypothèse de convexité sur les fonctions objectifs permettrait d'aller plus vite.

Une piste d'amélioration possible consisterait à tester d'autres méthodes de résolutions d'optimisations multi-objectifs afin de comparer avec d'autres mécanismes présentant la même flexibilité. Cependant, la tâche est complexe, car le nombre d'heuristiques disponibles dans la littérature est très élevé et implémenter ces heuristiques est long et met en jeu de nombreux paramètres de configuration rendant cette exploration difficile.

Une autre amélioration peut également être envisagée pour fournir un modèle de satisfaction des utilisateurs prenant en compte le nombre de requêtes reçues. Ainsi, le RPCA pourrait utiliser une modélisation de la satisfaction des utilisateurs tenant compte des popularités respectives des différentes URI.

Enfin, un RPCA pourrait également être étendu afin de gérer la fréquence d'envois des observations en provenance des nœuds. Dans certains LLNs, les nœuds envoient périodiquement des notifications à la passerelle afin qu'elle puisse redistribuer cette notification aux abonnés intéressés. N'avoir que la passerelle comme abonné allège la charge qu'un nœud doit gérer. Le RPCA pourrait configurer les nœuds pour réguler la fréquence d'envoi des notifications. L'objectif serait de trouver le bon compromis entre la satisfaction des utilisateurs qui veulent des informations régulièrement et les nœuds qui veulent envoyer aussi peu de notifications que possible.

Publications

— Rémy Leone, Paolo Medagliani et Jérémie Leguay. Optimizing QoS in Wireless Sensors Networks using a Caching Platform. *Sensornets 2013*, page 56, Barcelone, Espagne, Février 2013.

— Rémy Leone, Paolo Medagliani et Jérémie Leguay. Optimisation de la qualité de service par l'utilisation de mémoire cache 15èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications

Chapitre 5

Expériences automatisées et reproductibles pour LLNs

> Besides black art, there is only automation and mechanization.
>
> Federico Garcia Lorca

Sommaire

Ce chapitre présente Makesense, un framework permettant d'obtenir des expériences reproductibles, documentées et automatisables visant les LLNs. L'organisation du chapitre est la suivante :

La section 5.1 présente la reproductibilité d'expérience et la met en perspective avec la recherche sur les LLNs.

La section 5.2 présente comment la documentation et l'automatisation sont faites dans l'état de l'art, comment l'interaction avec les nœuds physiques d'une plateforme est classiquement faite et les limites de chacune de ces approches.

La section 5.3 présente Makesense et les choix techniques effectués pour l'utiliser efficacement dans le contexte des LLNs.

La section 5.4 conclut ce chapitre en justifiant l'intérêt d'un tel framework pour la communauté scientifique.

Introduction à la recherche reproductible dans les LLNs

Reproductibilité

Une expérience est définie comme une série d'actions ayant pour but de tester (confirmer ou infirmer) une hypothèse [67, 16]. Il y a trois éléments impliqués dans ce processus : le *laboratoire* qui correspond à l'environnement (conditions, scénarios, etc.) dans lequel l'expérience se produit, l'*expérimentateur* qui est la personne qui va faire l'expérience et le *dispositif* qui est étudié. Si une expérience peut être exécutée dans des laboratoires avec des expérimentateurs et des dispositifs tous différents, mais arriver aux mêmes conclusions alors l'expérience est dite reproductible.

Dans le cas des LLNs, une expérience peut consister à tester qu'un système (par exemple un cache) diminue le trafic qu'un LLN doit gérer. Dans ce cas, le laboratoire est l'endroit où cette expérience se produit, l'expérimentateur est l'agent humain ou automatisé effectuant l'expérience et le dispositif est le LLN (simulé, émulé ou réel) étudié.

Construire une expérience reproductible est à la base de la méthode scientifique. Pourtant mettre en place un environnement permettant de construire des expériences reproductibles est souvent perçu comme long et fastidieux. De plus, la reproductibilité est peu valorisée à court terme aussi bien au moment de juger un article pour le publier dans une revue qu'après sa publication [66, 196] et cela malgré son importance fondamentale [150]. Ce paradoxe a été relevé à de nombreuses reprises dans d'autres sciences et existe aussi en informatique [142].

Des dépôts de documents scientifiques rendent aujourd'hui possible l'hébergement des fichiers nécessaires au déroulement d'une expérience en plus de la publication en elle-même. Ainsi réduire la difficulté de mise en place d'expériences reproductibles est le verrou technique principal pour permettre sa diffusion et sa démocratisation.

Problématiques expérimentales des LLNs

Développer une expérience pour LLN peut être complexe en raison de la grande hétérogénéité des technologies mises en jeu (systèmes embarqués, transmissions de messages sur réseau, analyse de résultats) [19]. Ainsi des compromis doivent être trouvés entre le degré de réalisme souhaité et le temps de développement nécessaire pour mettre en place une expérience afin d'avoir des résultats pertinents au plus tôt.

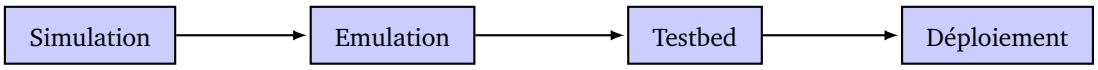


FIGURE 5.1 – Évolution d'une expérience vers le réalisme

La Figure 5.1 illustre les différents choix de niveaux d'abstraction de développement pour une expérience portant sur les LLNs. Il existe essentiellement 4 niveaux d'abstraction différents chacun ayant un différent compromis entre réalisme et facilité de développement.

Les simulateurs et émulateurs n'utilisent pas des nœuds physiques et permettent d'avoir dans un programme un fonctionnement modélisé d'un LLN afin d'illustrer son fonctionnement aussi rapidement que possible.

Vérifier des résultats de simulations sur des nœuds physiques apporte une confirmation aux résultats trouvés en simulation. Cependant effectuer des expériences sur un grand nombre de nœuds requiert de disposer physiquement de ces nœuds. Il peut être coûteux pour un laboratoire d'investir pour acheter, mettre en place et maintenir tous les nœuds requis pour une expérience donnée. Ainsi des plateformes publiques ("testbeds" en anglais) sont apparues afin de mutualiser un grand nombre de nœuds physiques entre plusieurs équipes de recherche [51, 68]. Ces plateformes mettent à la disposition de leurs utilisateurs des nœuds physiques avec des architectures matérielles variées sur lesquels un utilisateur déploie son système. Elles offrent des traces détaillées de l'expérience et permettent de déployer sur des nœuds représentatifs d'un capteur un code spécifique et de mesurer l'évolution de ce nœud au cours du temps [68].

Les sous-sections suivantes exposent plus en détail les différents compromis de chaque niveau et la difficulté de porter les développements faits à un niveau donné vers un niveau plus réaliste.

Expérience sur simulateur

Un simulateur permet d'avoir une modélisation du LLN sous la forme d'un programme qui une fois exécuté renvoie des résultats [171, 5].

L'atout d'un simulateur réside dans son environnement qui est totalement contrôlé, détaillé et sur la capacité de simuler des scénarios avec un grand nombre de nœuds sur des temps longs avec des temps courts de simulation [89]. Si les graines des générateurs de nombres aléatoires sont fixées, alors on a une reproductibilité complète de l'expérience si on ré-exécute le même programme avec les mêmes entrées deux fois de suite. En outre, le simulateur étant un système isolé, corriger le système et tester une nouvelle idée est relativement facile.

Cependant, les simulateurs font un certain nombre d'approximations 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. Ainsi les simulations ne correspondent pas fidèlement à un déploiement sur nœuds réels et des problèmes non prévus peuvent subvenir lorsqu'un déploiement réel se produit.

Expérience sur émulateur

Un émulateur permet de substituer une architecture matérielle physique par un logiciel. Les émulateurs sont activement utilisés afin de permettre un développement logiciel rapide qui ne nécessite pas d'installer un code sur un nœud physique à chaque changement [138].

Ainsi dans le cas d'une simulation, au lieu d'utiliser un programme ad hoc le simulateur utilise un émulateur d'architecture matérielle afin de disposer d'informations plus réalistes sur le fonctionnement d'un nœud comme des visualisations du cycle de veille des nœuds, de la consommation énergétique estimée de chaque nœud et de pouvoir inspecter le code s'exécutant sur un nœud durant une simulation. Émuler permet de tester un code réel, tandis que simuler oblige à utiliser une abstraction du code exécuté.

Cependant, un émulateur est soumis aux mêmes limitations qu'un simulateur, toutes les interactions (transmission, dérive d'horloge) sont modélisées par des logiciels et sont donc d'un réalisme limité.

Expérience sur plateforme

L'objectif des déploiements sur plateforme est de fournir des nœuds physiques réels aux expérimentateurs afin d'exécuter des expériences dans des conditions plus réalistes que celle d'un simulateur ou d'un émulateur.

L'utilisation des plateformes est répandue en recherche et de nombreux travaux s'appuient sur ces plateformes pour effectuer des expériences [76].

Cependant, effectuer une expérience sur plateforme présente des différences de fond par rapport aux expériences sur simulateur et émulateur. Le temps ne pouvant être accéléré comme sur un simulateur, des expériences longues sont nécessaires pour tester la fiabilité sur de grandes périodes de temps. Puisque les nœuds sont réels, ils peuvent tomber en panne et avoir des avaries, ainsi, il est impossible de garantir de manière systématique et déterministe, que deux expériences donneront exactement les mêmes résultats.

Dans le cas de plateformes mutualisées et distantes, il est nécessaire de partager les réservations des nœuds ainsi que de communiquer avec cette plateforme via des interfaces spécifiques ce qui rajoute un niveau de complexité à l'expérience. Enfin, les déploiements sur ces plateformes se font le plus souvent sur des nœuds à l'intérieur de bâtiment, statiques dans l'espace, dans des environnements contrôlés et offrant peu de variabilité. Ainsi, il n'est pas toujours possible de contrôler l'environnement radio, la position des nœuds et l'atténuation des signaux entre les différents nœuds.

Expérience en déploiement réel

Déployer une expérience sur nœuds réels permet d'avoir un système dans des conditions aussi réalistes et proches que possible que celles dites de "production" d'un système finalisé.

Le réalisme dans ce cas est très proche des conditions réelles d'un LLN déployé dans son environnement cible. Cette phase qui est généralement faite en phase de finalisation est plus coûteuse car il faut disposer des nœuds physiques, les installer dans une zone potentiellement large et subir les conditions logistiques de déploiement [194]. Dans ces conditions, modifier les nœuds peut être complexe si l'environnement est hostile ou si les nœuds ne peuvent pas être mis à jour facilement à distance.

Transitions entre niveaux

L'objectif d'une expérience est de se rapprocher autant que possible de la réalité afin de permettre de modéliser des problèmes et d'y apporter des solutions. Cette progression vers le réalisme peut être accélérée par la réutilisation des systèmes développés lors d'étapes préalables.

Passer de l'étape de simulation à l'émulation permet de gagner en réalisme et de modéliser plus finement un LLN car une partie des contraintes mémoires des architectures matérielles des nœuds sont prises en compte grâce à l'émulateur. L'analyse des données et leur visualisation peuvent être conservées, car une partie des sorties de l'expérience (par exemple les logs séries) peuvent être au même format et analysées de la même façon.

Le passage de l'émulation à la plateforme permet de gagner en réalisme, car les contraintes des nœuds physiques apparaissent. Les images binaires produites lors du prototypage avec un émulateur peuvent être conservées, ainsi il n'est pas nécessaire de les réécrire. De plus, le code écrit pour analyser les résultats peut être également conservé, car les sorties séries peuvent être au même format.

Enfin, le passage de la plateforme au déploiement réel met en jeu des contraintes qui sont spécifiques à un déploiement donné et à sa finalisation, ainsi cette difficulté souvent logistique ne peut être évitée et dépend de chaque contexte. Cependant, le binaire faisant fonctionner les nœuds et l'analyse des résultats conçus précédemment peuvent être conservés.

On peut observer que la progression vers le réalisme est un processus itératif et qu'une partie des développements peuvent être conservés d'un niveau à l'autre afin d'économiser les coûts de mise en place. Ainsi un framework permettant de documenter, d'automatiser et de réutiliser autant de fonctionnalités que possible entre les différents niveaux permet aux utilisateurs de se concentrer sur

les verrous technologiques propres à chaque niveau au lieu de ré-implémenter l'ensemble de leur développement à chaque fois.

État de l'art sur les outils de gestion d'expériences

Gérer une expérience sur des nœuds physiques ou émulés met en jeu de nombreuses étapes comme la préparation des nœuds, l'exécution et le traitement des données. L'intégration de toutes ces étapes est longue et fastidieuse, ainsi il est nécessaire de la documenter et lorsque cela est possible de l'automatiser.

Documentation

Les logiciels et plateformes utilisés par les LLNs sont relativement nouveaux, ainsi, il est particulièrement important que leur documentation soit aussi claire et tenue à jour que possible afin de diffuser efficacement leur usage. Cette documentation est actuellement essentiellement fournie par des didacticiels ad hoc et des wikis qui sont édités par la communauté des utilisateurs. C'est notamment le cas des plateformes Indriya et FIT-IoT-lab [68, 50] et des systèmes d'exploitation usuels tels que Contiki [53], OpenWSN [193] ou encore RIOT [14]. Cette approche est choisie, car elle est facile à mettre en place et permet de rapidement partager une expérience avec les utilisateurs, mais a le défaut de pouvoir devenir obsolète si elle n'est pas tenue à jour.

Une autre approche consiste à documenter un projet de manière systématique par l'utilisation d'un logiciel de documentation comme Doxygen [185] ou Sphinx [23]. Cependant cette approche repose sur une idée de projet unique et cohérent, or les expériences sur les LLNs mettent en jeu de très nombreux sous-projets hétérogènes dans des langages parfois différents rendant cette approche peu efficace.

Enfin, une approche alternative à la documentation consiste à intégrer du code exécutable dans le même document qui est utilisé pour décrire une expérience. Ces documents enrichis sont souvent désignés par le terme de *notebook* et sont couramment utilisés dans certaines communautés scientifiques et pédagogiques [201, 165]. Cependant l'utilisation de ces notebooks est encore restreinte dans la communauté de recherche sur les LLNs.

Automatisation

Automatiser une expérience permet de la rendre reproductible, documentée et de la partager plus facilement, car toutes les étapes requises sont décrites dans le programme qui l'automatise [109, 98]. De plus, l'automatisation d'un système permet de fournir des résultats cohérents d'une exécution d'expérience à l'autre et d'accélérer la prise en main d'un débutant, car toutes les étapes requises sont spécifiées. En outre, corriger une erreur dans un procédé automatisé permet de l'éviter définitivement alors qu'une correction manuelle appliquée de manière ad hoc doit être effectuée de nouveau lorsque le contexte de l'expérience change.

Afin de décrire les dépendances entre les différentes étapes de l'expérience [1], des outils génériques tels que `make` sont généralement utilisés pour garantir que les dépendances d'une étape donnée de l'expérience sont à jour [63]. Cependant, les vérifications préalables au lancement des étapes telles que les dépendances logicielles et matérielles sont rarement documentées ou testées lors de leur automatisation. Or, ne pas tenir compte de ces différentes dépendances peut provoquer des erreurs subtiles à détecter notamment dans des logiciels en développement actif comme les systèmes d'exploitation des nœuds capteurs d'un LLN.

1. Par exemple : compiler les nœuds avant de les émuler.

Déploiement d'expérience sur nœuds réels

Un testbed met en jeu de multiples ressources pour des utilisateurs multiples, ainsi la gestion des ressources disponibles et leur réservation est une question fondamentale pour ces plateformes. L'interface avec les testbeds est une question étudiée et de nombreux outils sont disponibles dans l'état de l'art pour s'interfacer avec eux aussi efficacement que possible [26]. Certains testbeds utilisent des frameworks spécifiques comme Orbit Management Framework (OMF) et son langage de description d'expériences OMF Experiment Description Language (OEDL) qui peuvent être utilisés pour décrire une expérience et déclencher des actions en fonction d'événements [155]. Cependant cette approche est coûteuse en temps, car elle requiert d'apprendre un Domain Specific Language (DSL) qui repose sur un formalisme fixe et qui ne peut pas être réutilisé dans d'autres contextes. Ainsi des approches plus génériques et indépendantes d'un langage donné sont plus adaptées pour favoriser la diffusion d'une expérience par exemple par l'utilisation d'une Application Programming Interface (API).

De plus, les frameworks permettant d'utiliser des testbeds disponibles dans la littérature ne proposent des abstractions que pour la réservation et le lancement d'expériences [13, 18]. Les interfaces proposées ne gèrent pas par exemple les processus et leur relance qui doivent être gérés par l'utilisateur. Ainsi il est nécessaire pour un utilisateur de surveiller l'évolution d'une expérience pour réagir aux pannes éventuelles. De plus, pour un utilisateur débutant, les multiples erreurs pouvant survenir sont parfois difficiles à interpréter quand peu d'informations sont disponibles pour aider.

Conclusion intermédiaire

Comme vu dans cette section, l'état de l'art offre des solutions pour chacun des problèmes individuellement cependant leur intégration au sein d'un seul framework cohérent est encore manquant.

Les descriptions de protocoles expérimentaux peuvent être enrichies via un mécanisme de notebook pour fournir à la fois une description et du code au sein d'un même fichier.

L'automatisation est nécessaire pour garantir la consistance et la rapidité de mise en place, mais doit être couplée à la vérification des dépendances et intégrée avec la documentation afin de garantir que le code d'une expérience est exécuté dans un contexte défini et que toutes les dépendances sont satisfaites.

Enfin, l'utilisation des testbeds est utile pour avoir des expériences sur nœuds physiques et il faut privilégier leur intégration au sein de mécanismes généraux comme des API afin d'éviter des DSL qui forcent l'utilisation d'un formalisme spécifique.

Makesense

Makesense est un framework produit dans le cadre de cette thèse permettant d'obtenir la *documentation*, l'*automatisation* et la *reproductibilité* d'une expérience. Ce framework vise à réduire le temps nécessaire pour mettre en place une expérience reproductible en intégrant les différentes étapes d'une expérience dans un seul document enrichi afin d'exécuter une expérience aussi bien sur des nœuds simulés, émulés ou réels. Makesense intègre plusieurs outils afin d'aboutir à cet objectif.

Utilisation d'un notebook pour contenir la documentation et le code d'une expérience

Makesense utilise un notebook pour contenir à la fois la documentation d'une expérience et le code permettant de l'exécuter. L'apport de fond d'un notebook se situe dans la juxtaposition de la documentation enrichie et du contexte d'exécution commun à l'ensemble du notebook. La description d'une expérience dans un notebook est présentée de la même façon que dans un didacticiel, des étapes se suivent dans un ordre précis pour aboutir à un résultat. Chaque étape peut être rejouée

indépendamment, ainsi le coût pour rejouer une étape se limite a l'étape elle-même et pas à ses dépendances qui sont gardées en mémoire.

Utilisation d'un langage et d'un contexte communs

Utiliser des outils hétérogènes et incapables de communiquer efficacement entre eux augmente la charge de développement et le temps requis pour intégrer ces différents composants [175]. Ainsi Makesense utilise un seul langage et partage un contexte d'exécution entre toutes les étapes afin de simplifier la compréhension d'une expérience et le partage des données.

Utilisation d'outils généraux et communs à plusieurs champs de recherche

Développer un outil spécifique aux LLNs limiterait d'emblée sa portée. En outre, les processus de gestion d'expériences ont de nombreuses similarités avec des approches génériques de développement logiciel. Makesense utilise des bibliothèques logicielles scientifiques et généralistes largement diffusées pour effectuer une expérience. Ce choix permet d'éviter l'obsolescence des dépendances, car elles reposent sur des communautés actives et visent à offrir un pont entre les communautés scientifiques et celles liées au développement logiciel.

Approvisionnement d'expérience par l'utilisation de gabarits

Les expériences sur les LLNs mettent en jeu de nombreux fichiers (code source des nœuds, configuration d'un simulateur, etc.) qui peuvent être réutilisables d'une expérience à l'autre, notamment dans les cas de campagnes d'expériences. Makesense génère ces fichiers de manière expressive en utilisant des gabarits de fichiers pour générer dynamiquement l'ensemble des fichiers nécessaires à une expérience donnée. Ainsi une campagne d'expérience composée d'expériences paramétrables peut être conçue rapidement à partir des gabarits et des paramètres spécifiques de cette expérience.

Réutilisation des processus d'exploitation sur différents niveaux d'abstraction

Une expérience, qu'elle se passe sur nœuds simulés, émulés ou physiques, conserve d'un niveau d'abstraction à l'autre des processus et des fonctions communes (par exemple l'analyse des données et leur mise en forme).

Makesense permet de documenter et d'automatiser ces processus communs afin de pouvoir les réutiliser autant que faire se peut d'un niveau d'abstraction à l'autre. Ainsi les transitions entre les différents niveaux sont plus rapides, car les traitements communs sont conservés et peuvent être appliqués en séquence.

Utilisation d'outils de gestion de révisions et d'intégration continue

Une expérience évolue au cours du temps, ainsi il est essentiel de garder en mémoire son évolution et de tester en permanence que les révisions apportées ne cassent pas les résultats précédemment obtenus et que les dépendances sont satisfaites. Ainsi Makesense utilise un gestionnaire de révision et un service d'intégration continue afin de conserver toute l'évolution du développement d'une expérience et garantir que chaque nouvelle révision d'une expérience n'introduit pas d'erreurs.

Licence et démonstration

Afin de favoriser son utilisation, Makesense est disponible sous licence Apache sur Github [2] et une démonstration du déroulement d'une expérience est également disponible [3].

Présentation d'une expérience typique sur les LLNs

Une expérience portant sur un LLN commence le plus souvent par la génération des différents composants nécessaires à sa mise en place : binaire faisant fonctionner les nœuds, préparation et configuration des outils annexes (émulateurs, simulateurs, etc.). Lorsque ces fichiers sont prêts, ils sont placés sur les machines adéquates : flashage de nœuds physiques, déplacement du programme de simulation vers un nœud de calcul, etc. Puis l'expérience est lancée, plusieurs processus concurrents sont mis en œuvre pour produire des résultats bruts : démarrage des nœuds physiques, lancement d'un simulateur, collecte des résultats, injection de trafic, etc. Lorsque l'expérience est terminée, les traces physiques sont analysées pour fournir des résultats quantitatifs et qualitatifs au sujet de l'expérience.

Toutes ces étapes mettent en jeu un grand nombre de technologies hétérogènes et sont donc difficiles à maintenir et documenter efficacement.

Le but de Makesense est de documenter, automatiser et rendre reproduire des expériences scientifiques et vise tout particulièrement les expériences sur les LLNs. Les sous-sections suivantes décrivent comment Makesense apporte une solution aux problématiques préalablement énoncées.

Documentation d'une expérience

Une expérience sur un LLN intègre de nombreuses étapes comme la construction des images binaires qui vont être flashées, le lancement des processus, le traitement des données, etc. Des erreurs peuvent arriver à chaque étape, ainsi leur intégration est coûteuse notamment dans le cas où l'expérimentateur a peu d'expérience au sujet des LLNs et des logiciels mis en jeu.

Une approche classique de la documentation d'une expérience consiste à avoir la documentation, le code et les résultats obtenus dans des fichiers distincts. Cette approche n'offre aucune garantie de filiation entre la documentation, le code et les résultats obtenus notamment dans le cas où la documentation et le code sont encore activement édités. De plus, dans le cas où il ne s'agit que d'un seul code monolithique, il est nécessaire de le ré-exécuter entièrement pour avoir ses résultats, ceci peut s'avérer coûteux dans le cas où des traitements longs sont ré-exécutés inutilement.

Makesense documente ces étapes, en utilisant des notebooks afin d'avoir d'un seul référentiel pour le code et la documentation que l'expérience se passe sur émulateur ou sur testbed.

Introduction sur les notebooks

Un notebook est un document interactif qui permet de ré-exécuter simplement toutes les étapes d'une expérience afin d'obtenir ses résultats dynamiquement. Il est composé d'un noyau effectuant les calculs et de cellules de texte enrichies pouvant être affichées avec une mise en page avancée.

L'annotation et la modification des textes en marge des résultats permettent d'avoir une façon beaucoup plus lisible de partager des résultats et de savoir comment ils ont été obtenus. En outre, modifier la documentation du processus expérimental est beaucoup plus naturel, car elle se trouve à côté du code permettant de l'obtenir et au sein du même document.

Un notebook, une fois lancé, peut être vu et utilisé via une interface graphique comme montré sur la Figure 5.2 qui met en jeu une interface web. Cette figure met en jeu quatre cellules, la première

2. https://github.com/sieben/makesense
3. https://travis-ci.org/sieben/makesense

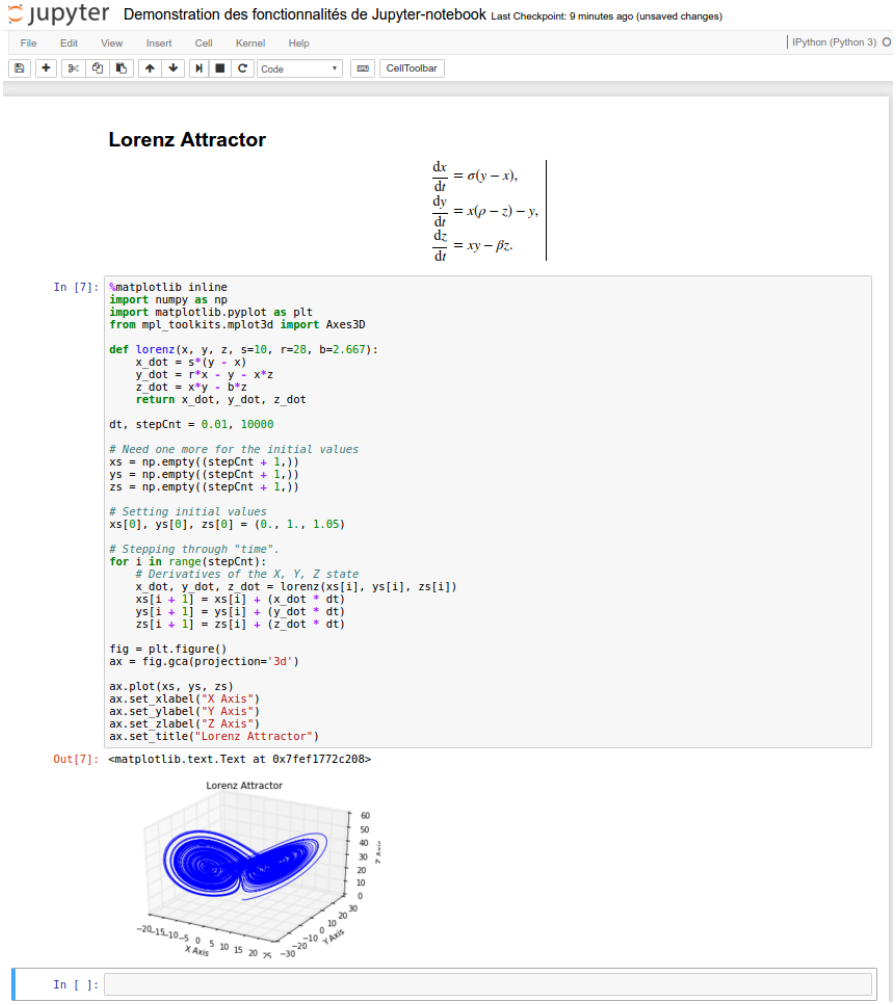


FIGURE 5.2 – Capture d'écran d'un notebook ouvert fonctionnant en local et consulté par interface web

représente un rendu de texte enrichi avec un titre mis en gras et des formules LaTeX rendues et centrées. Lorsqu'une cellule de texte est rendue, le texte et les formules mathématiques au format LaTeX ou Markdown contenus dans cette cellule sont rendus à l'utilisateur permettant d'avoir une mise en page avec des listes, des sections, des images et des formules mathématiques.

La seconde cellule est une cellule de code : elle contient du code et des "built-in magic commands" introduites par le signe % au début de la cellule et servant à modifier dynamiquement le comportement du notebook. Lorsqu'une cellule de code est rendue, le code est exécuté dans le noyau qui est un contexte commun pour toutes les autres cellules de code du notebook. De plus, les cellules de code peuvent être ré-exécutées indépendamment les unes des autres permettant par exemple de séparer les calculs longs et coûteux des calculs plus courts d'exploration ou de visualisation de données.

Enfin, la troisième cellule montre le résultat de l'exécution du code contenu dans la seconde cellule sous forme graphique. La quatrième cellule est une cellule de code disponible pour être éditée et exécutée de manière interactive.

L'utilisation des notebooks dans les logiciels scientifiques est ancienne et cette technologie est

devenue suffisamment mature pour être utilisée aussi bien en recherche qu'en enseignement [71, 201, 178, 144].

Parmi l'ensemble des notebooks libres disponibles, on peut citer Wakari [183], Sage [177], Apache Zepplin [123] ou encore IPython [143].

Makesense intègre dans un notebook l'ensemble des traitements, paramètres, fonctions et la documentation du protocole expérimental d'une expérience sur les LLNs. Makesense peut fonctionner avec plusieurs formats de notebook différents tant qu'ils supportent certaines fonctionnalités.

Jupyter [22] est un projet proposant une suite d'outils libres pour l'informatique interactive et parallèle scientifique. Ce projet est une évolution du projet IPython [143] qui a été étendu pour donner le projet Jupyter. Le Jupyter Notebook est le format de notebook fournit par le projet Jupyter combinant code, texte, images, expressions mathématiques et des fonctions interactives au sein d'un même document qui est encodé dans un format texte documenté [152]. Ces notebooks peuvent être lancés dans des interfaces graphiques ou web qui offrent un rendu graphique et un contexte d'exécution commun par l'utilisation d'un noyau. Jupyter dispose d'une communauté d'utilisateurs et de développeurs large et dynamique ainsi que de nombreux ouvrages documentant son fonctionnement [127].

Les paragraphes suivants détaillent les différents critères et justifient pourquoi Makesense utilise dans cette thèse la plateforme Jupyter et son format de notebook afin de documenter et effectuer des expériences.

Propriétés recherchées pour un notebook

Format texte et ouvert

Reproduire une expérience doit avoir aussi peu de barrières que possible, or acheter une licence ou un abonnement pour utiliser un logiciel propriétaire représente une barrière non négligeable. Ainsi seuls les formats ouverts et pouvant être utilisés par des solutions libres sont pertinents pour un notebook visant à être utilisé par un public large.

Utiliser un format texte permet en outre de visualiser facilement les différences entre deux versions d'un même notebook par des outils de comparaisons de documents textes classiques ce qui permet de faciliter l'acceptation des contributions extérieures. De plus, la gestion des révisions est nécessaire pour la mise en place d'une intégration continue efficace qui sera introduite dans la section 5.3.4.1.

Jupyter Notebook utilise un format texte ouvert et documenté sous la forme d'un Javascript Serial Object Notation (JSON) [152] et répond donc à ce besoin fonctionnel.

Flexibilité dans le choix du langage de programmation

Les préférences de langage de programmation peuvent varier en fonction d'une expérience et des préférences d'un utilisateur. La notion de notebook est indépendante du langage qu'elle utilise, car c'est avant tout un format contenant du texte enrichi et de la mise en page de cellules. Ainsi un notebook idéal doit être indépendant d'un langage de programmation donné pour laisser autant de flexibilité que possible à un utilisateur.

Le format de notebook de Jupyter Notebook est découplé de son noyau d'exécution, ainsi il existe plus d'une soixantaine de noyaux différents, écrits dans plusieurs langages de programmation [22]. Les critères de choix d'un moteur sont relatifs à la disponibilité des bibliothèques logicielles, la stabilité du moteur et les propriétés du langage.

Afin de bénéficier de la plus grande stabilité et d'une large gamme de bibliothèques logicielles scientifiques, Makesense utilise `ipykernel` qui est le noyau historique de référence de Jupyter codé en Python. De plus, l'utilisation de Python permet d'interfacer facilement Makesense avec la plateforme FIT-IoT-lab (5.3.6) par l'utilisation de la bibliothèque officielle d'IoT-lab qui est codée en Python.

Support d'exports et de rendu

Un notebook utilisé par Makesense doit pouvoir répondre à deux besoins relatifs au partage et l'export d'une expérience afin de pouvoir couvrir un large éventail d'utilisation et être flexible sur les préférences de leurs différents utilisateurs.

Le premier consiste à pouvoir être exportable vers des fichiers statiques (Portable Document Format (PDF), page web, etc.) afin que les résultats d'une expérience puissent être partagés sans nécessiter une installation de dépendances et l'exécution des cellules.

Jupyter-notebook dispose d'un grand nombre d'exports et un rendu automatique des notebooks est supporté par plusieurs plateformes en ligne comme Github [4] et nbviewer [5]. Ainsi un notebook peut être rendu directement sur la plateforme Github pour avoir un rendu similaire à celui de la Figure 5.2 en ligne et sans installation. Le rendu sur ces plateformes est fait sur une page web statique pouvant être imprimée ou partagée avec des relecteurs.

Le second besoin consiste à pouvoir extraire les cellules de code vers un programme exécutant l'ensemble de l'expérience. Cette approche est pertinente pour disposer d'un programme pouvant ré-exécuter l'expérience et produire ses résultats et où le rendu du texte n'est pas souhaité.

Jupyter-notebook supporte ces deux types d'exports et permet ainsi à Makesense de répondre à ce besoin fonctionnel.

Automatisation d'une expérience

Automatiser les tâches d'un processus est essentiel pour s'assurer qu'aucune étape n'a été oubliée et qu'elles sont toutes explicitées. Dans le cas de développement d'expériences collaboratives, automatiser permet d'éviter les configurations spécifiques d'une machine, et favorise la synchronisation du code et des résultats.

Makesense permet d'obtenir à partir du notebook d'une expérience son automatisation et de garantir sa reproductibilité.

Dans le cas de simulations, l'automatisation a pour but d'assurer que la construction, l'exécution et l'obtention des résultats sont valides et reproductibles. Lorsque des paramètres aléatoires sont utilisés, pour par exemple injecter un bruit aléatoire sur un canal, un simulateur est complètement déterminé dès lors que la graine du générateur de nombres aléatoires est fixe et que les paramètres d'entrées sont constants.

Dans le cas des déploiements réels, les nœuds physiques utilisés pour réaliser une expérience peuvent changer en raison de pannes ou de problèmes d'exploitations, ce qui rend l'automatisation plus délicate, mais encore possible. En outre, dans le cas d'expériences se déroulant sur une plateforme publique, une exécution peut être perturbée par les expériences gérées par d'autres utilisateurs par exemple pour les conditions radio.

Comme vu dans la section 5.3.2.2, il est possible de convertir un notebook en un programme par extraction des cellules de code qu'il contient. L'exécution de ce programme permet de refaire étape par étape une expérience automatiquement.

Découpage d'une expérience en étapes

Une expérience peut être découpée en différentes étapes qui se suivent dans un ordre logique. Le schéma 5.3 illustre comment les différentes étapes d'une expérience typique à propos des LLNs se suivent.

4. https://github.com/sieben/makesense/blob/master/demo.ipynb
5. http://nbviewer.jupyter.org

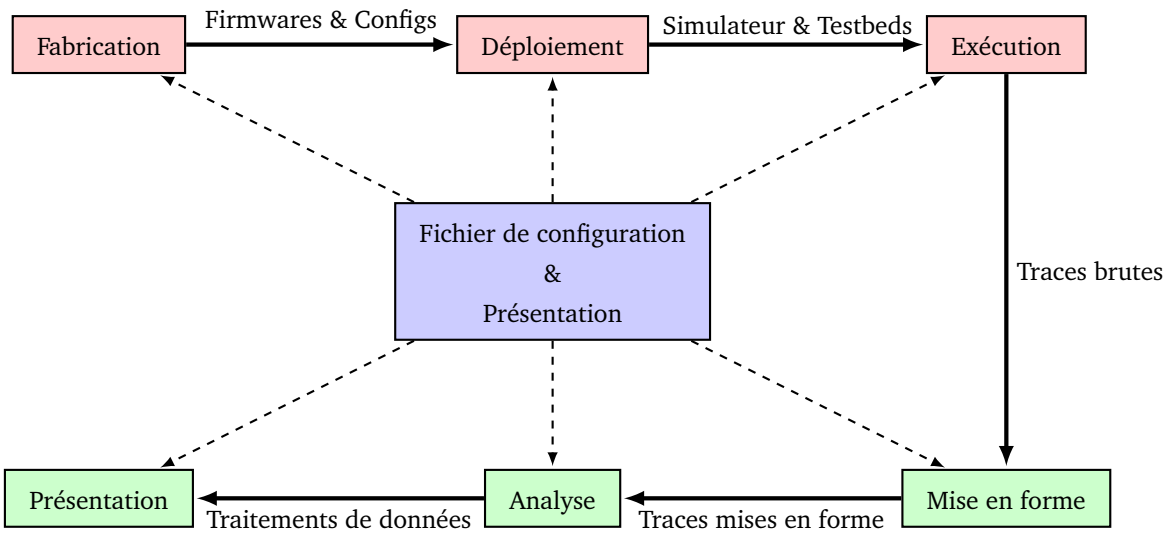


FIGURE 5.3 – Découpage des étapes d'une expérience par Makesense

La *fabrication* produit tous les binaires et fichiers de configurations initiaux qui sont nécessaires à une expérience. La phase de *déploiement* déplace ces fichiers sur les bonnes machines. La phase d'*exécution* lance et maintient les processus nécessaires à une expérience et produit les traces brutes qui correspondent aux grandeurs mesurées. La phase de *mise en forme* "nettoie" des traces brutes afin de les rendre plus simples à analyser et exploiter. La phase d'*analyse* traite ces données mises en forme afin de produire des analyses qualitatives et quantitatives sur les phénomènes observés. Enfin, la phase de *présentation* fournit les représentations graphiques qui seront le plus souvent mises dans des articles ou partagées avec d'autres collaborateurs.

Chacune des cellules de code utilisée peut être précédée ou suivie de plusieurs cellules de textes enrichis pour décrire ce qu'elle contient ou apporter des précisions sur son fonctionnement. Les résultats intermédiaires des cellules de code peuvent être conservés dans le contexte commun à toutes les cellules du notebook, ce qui permet de réutiliser des résultats et au besoin d'ajouter une nouvelle cellule pour explorer une nouvelle idée.

Expérience séquentielle et script d'exécution

Comme vu dans la section 5.2.2, il est nécessaire d'utiliser un outil de gestion de dépendance de tâches pour s'assurer que toutes les dépendances d'une tâche sont exécutées. Or comme l'illustre le schéma 4.1, une expérience typique est bâtie autour d'un succession séquentielle d'étapes. Ainsi l'organisation du code d'un notebook en succession d'étapes est acceptable pour ce type d'expérience rendant les outils génériques de gestion de dépendances de tâches tel que `make` non nécessaire par rapport aux besoins.

Il est donc possible de mettre chacune des étapes d'une expérience dans une cellule spécifique, ce qui permet de rendre une expérience modulaire, car on peut obtenir le résultat d'une étape donnée en exécutant seulement une fois ses dépendances et en conservant les résultats intermédiaires. Ce découpage permet de ne pas imposer de formalismes lourds sur une expérience et permet d'avoir une grande flexibilité sur son organisation.

Ce flot séquentiel est également adapté pour fournir un programme validant une par une toutes les étapes pour aboutir à la production des résultats. Ce programme séquentiel est obtenu par l'extrac-

tion des cellules de code du notebook qui fournissent un programme pouvant reproduire l'exécution complète de l'expérience. Ainsi il est possible de tirer parti de ce programme pour tester de manière systématique que l'expérience fonctionne d'une révision à l'autre.

Garantie de reproductibilité d'une expérience

Une expérience doit évoluer au cours du temps afin de corriger ses erreurs et obtenir de nouveaux résultats. Cependant cette évolution ne doit pas se faire au détriment de la stabilité des résultats précédemment obtenus, car un processus expérimental se construit sur des acquis pour ensuite les étendre.

Un processus de développement logiciel efficace doit garantir une conservation des différentes révisions afin de revenir en arrière si une erreur est introduite ou bien de gérer les révisions concurrentes de plusieurs contributeurs. Ainsi des logiciels spécifiques ont été introduits pour tenir compte de ces révisions et les gérer efficacement. Les gestionnaires de versions sont aujourd'hui une fondation du développement logiciel et sont largement utilisés [139].

Makesense utilise un gestionnaire de révisions afin de garder en mémoire les différentes révisions d'un notebook. L'observation des différences entre les différentes révisions est importante pour comprendre comment chaque révision modifie un notebook donné. `git` est le gestionnaire de révision de référence de Makesense car il permet de disposer depuis la plateformes Github d'un rendu automatique du notebook pour toutes ces révisions successives [119, 45]. Cependant, Makesense est complètement agnostique sur le gestionnaire de version utilisé et ne l'utilise que pour mettre en place des mécanismes d'intégration continue.

Intégration continue

L'obsolescence d'un programme peut arriver quand aucun test n'est présent pour s'assurer qu'il fonctionne ou non au cours de ses révisions successives. Pour éviter cette obsolescence, il est nécessaire d'utiliser un processus systématique qui va vérifier à chaque révision que les résultats précédemment obtenus sont toujours valides. Comme vu dans la section 5.3.3.2, un notebook peut être transformé en script qui va exécuter séquentiellement une expérience pour en fournir les résultats.

L'*intégration continue* est un ensemble de pratiques utilisées en production de logiciel qui consiste à vérifier que chaque modification apportée à un programme ne modifie pas son fonctionnement et qu'aucune régression fonctionnelle n'est introduite [57]. Son principal but est de détecter les problèmes et les erreurs afin de les corriger au plus tôt et de raccourcir les cycles de développement. Massivement pratiquée par les acteurs principaux du développement logiciel elle a engendré de nouveaux courants et méthodologies logicielles comme le Test Driven Development (TDD) [20] qui ont prouvé leur efficacité pour réduire le nombre d'erreurs au cours du développement d'un logiciel [125].

L'intégration continue est déclenchée quand une proposition de changement [6] est poussée vers le dépôt gérant les versions du notebook. Une copie de la version révisée est envoyée sur un serveur qui va créer l'environnement pour le notebook, extraire les cellules de code puis exécuter le programme engendré. Une fois l'exécution terminée et sans erreur, on a la garantie que le changement n'a pas engendré une erreur sur un résultat existant [7]. Une fois que le résultat est conforme aux tests demandés, il peut être intégré à la branche principale de développement et le processus se répète pour tout nouveau changement proposé.

Cette méthodologie de développement logicielle combinée a un script extrait d'un notebook permet de garantir que l'expérience contenue reste reproductible.

6. "pull request" dans le cas où on utilise `git`.
7. On parle de test de non-régression.

L'intégration continue dans le cas d'expérience sur un LLN garantit que toutes les dépendances requises pour créer une expérience, l'exécuter et exploiter les résultats sont documentées, installées avec une intégration testée et fonctionnelle.

Makesense versionne l'évolution du notebook contenant l'expérience de cette section en utilisant `git` et en l'hébergeant sur Github. Lorsqu'une nouvelle révision est proposée (pour ajouter une étape, changer une variable), une proposition de révision est envoyée pour être relue par les propriétaires du dépôt hébergeant le notebook. Lorsque la proposition de révision est initiée, une copie de la version modifiée est envoyée vers le serveur d'intégration continue qui déterminera si cette révision est acceptable. Cette vérification est obtenue en procédant à l'extraction des cellules de code du notebook et l'exécution du script obtenu. Une fois cette vérification faite, la révision peut être intégrée au dépôt du notebook et ce cycle se répète à chaque révision proposée.

Approvisionnement d'une expérience

L'approvisionnement d'une expérience consiste à préparer tous les fichiers et systèmes nécessaires au déroulement d'une expérience. Les expériences sur LLNs mettent en jeu des nœuds ayant des configurations spécifiques par exemple sur la fréquence d'envoi d'un message. Configurer chaque nœud de manière ad hoc n'est pas envisageable à mesure que le nombre de nœuds et les paramètres mis en jeu grandissent. Afin de réaliser des expériences à grande échelle, le langage utilisé pour décrire l'expérience doit être suffisamment souple et expressif pour initialiser facilement chaque nœud à partir d'un modèle de base.

La mise en œuvre de cette phase de configuration dans Makesense consiste à créer à partir de gabarits de base (appelé "template" en anglais), les fichiers qui seront utilisés et compilés au cours de l'expérience. Chaque fichier nécessaire à l'expérience va être construit à partir d'un gabarit de base qui va être rempli par un moteur de gabarit (aussi appelé "templating engine" en anglais) avec les variables qui lui sont propres. Ce type de pratique par gabarit a déjà été utilisé et validé à large échelle dans le contexte de configuration de serveurs et de configuration de serveurs [80] ainsi son utilisation dans le contexte de la configuration des nœuds capteurs se rapproche de cette utilisation.

Dans le cas typique des expériences portant sur un LLN, cette étape de préparation compile les systèmes d'exploitation pour les nœuds contraints et produit tous les fichiers de configuration nécessaires à la simulation. Dans le cas du binaire exécuté par les nœuds, il est obtenu pour différentes architectures [8].

Une simulation dans Cooja requiert la création d'un fichier principal qui va être utilisé pour placer les nœuds, les démarrer avec le bon binaire, interagir avec eux, configurer leur portée de transmission ou la graine des générateurs de nombres aléatoires utilisée pendant la simulation. Ainsi, Makesense fournit un gabarit de fichier de configuration qui est peuplé dynamiquement et enregistré pour être utilisé ensuite. Le rendu de la topologie est fait automatiquement et les binaires sont compilés correctement avec leurs variables propres et sont affectés aux nœuds correspondants dans le fichier de configuration de la simulation.

Déploiement - Exécution et récupération des traces

L'objectif des phases de déploiement, d'exécution et de déplacement des traces est de produire une exécution de l'expérience visée et de déplacer toutes les traces pertinentes de cette expérience vers une destination spécifique pour y être analysées. Cette phase est délicate car elle met en jeu de nombreux processus. Elle est documentée et automatisée par Makesense afin de fournir autant

8. L'émulation dans COOJA s'effectue sur l'architecture MSP430 et les nœuds FIT-IoT-lab utilisent l'architecture ARM dans le cas des Cortex M3.

que faire se peut une reproductibilité de l'expérience qu'elle se passe dans un simulateur ou sur plateforme.

Déploiement

La phase de déploiement déplace les fichiers produits lors de l'approvisionnement vers la machine où ils seront utilisés.

Makesense déploie une expérience vers un serveur ou une plateforme en utilisant les API ou un accès à distance automatisable. Ainsi toute plateforme ou machine utilisant un accès à distance automatisable est utilisable par Makesense.

Dans le cas de l'exécution d'une simulation sur la même machine, ce déploiement correspond à une simple copie des fichiers pertinents. Cependant, dans le cas de simulations devant être exécutées sur des grilles de calcul, il est nécessaire de mémoriser sur quels serveurs l'expérience a été déployée afin d'en récupérer les résultats.

L'utilisation d'un testbed permet de confirmer les résultats d'un simulateur par une exécution de l'expérience sur des nœuds physiques. Dans le cas de testbeds mutualisés, il faut gérer des réservations de nœuds qui sont potentiellement accessibles par tous les utilisateurs de la plateforme et qui peuvent également tomber en panne. Ainsi il est nécessaire de maintenir dynamiquement une table de correspondance entre les nœuds réservés et le code qui y est exécuté. Automatiser ce processus de déploiement permet de choisir dynamiquement les nœuds mis en jeu lors d'une expérience (par exemple les moins chargés, les mieux placés, etc.).

Plusieurs testbeds publics tels que Indriya [50] ou bien IoT-lab [68] sont disponibles afin d'exécuter une expérience sur des nœuds physiques. Makesense supporte IoT-lab par l'utilisation de la bibliothèque officielle pour réserver et démarrer des expériences. La plateforme IoT-lab a été choisie en raison de la variété des nœuds physiques proposés, du support des systèmes d'exploitation de référence sur ses nœuds et de son processus de réservation et d'accès à distance facilement automatisable. IoT-lab met à disposition de ses utilisateurs, sur plusieurs sites physiques ayant la même interface, des nœuds physiques supportant IEEE 802.15.4 représentatifs de nœuds capteurs. Pour fonctionner, IoT-lab utilise, entre autre, des serveurs (nommés "front-end") sur lesquels des utilisateurs peuvent se connecter par Secure Shell (SSH) et lancer des commandes visant à interagir avec les nœuds physiques. Makesense utilise la bibliothèque officielle d'IoT-lab afin de faire les réservations et d'envoyer les binaires utilisés par les nœuds physiques. Dans le cas où d'autres programmes doivent être déployés sur les serveurs d'IoT-lab, Makesense y accède par SSH afin d'y envoyer les fichiers requis.

Exécution

Lors de l'exécution d'une expérience, il est courant d'avoir plusieurs processus lancés en parallèle pour effectuer différentes fonctions. Parmi ces processus, certains vont par exemple, générer les traces brutes qui vont être utilisées par la suite pour analyser le déroulement d'une expérience.

Ces processus sont parfois nombreux, s'exécutent avec différents privilèges, sur différentes machines et dans un certain ordre. En outre, il faut s'assurer qu'une fois lancés, ces processus continuent de fonctionner correctement durant toute l'expérience. Gérer manuellement ces processus comporte plusieurs risques, d'une part, il est possible d'en oublier un ou bien de les lancer dans le désordre causant l'échec de l'expérience si ces processus sont critiques. De plus, dans le cas où les expériences durent longtemps, il est nécessaire de maintenir une surveillance continue de l'expérience et de chacun de ces processus.

Makesense permet de documenter le lancement de ces processus en les mettant dans des objets qui sont gérés de manière automatisée. Ainsi les processus sont systématiquement lancés et dans le

même ordre permettant d'éviter les écueils d'un lancement manuel. D'autre part, il est possible d'avoir au cours de l'exécution le statut d'un processus et le relancer le cas échéant.

Récupération des traces

Une expérience, qu'elle se produise dans un simulateur ou sur nœuds physiques, génère de nombreuses données brutes comme des fichiers Packet CAPture (PCAP) [126] qui décrivent le trafic réseau ou encore des journaux d'événements (appelés "logs" en anglais).

Afin de simplifier les traitements, il est pratique de rassembler toutes ces traces sur une seule machine qui va être utilisée pour traiter ces données. Ainsi il est nécessaire d'agréger les différents fichiers de données en provenance de sources multiples vers une seule destination. Automatiser ces traitements permet de garantir qu'aucun de ces fichiers n'est oublié, de documenter ceux qui seront traités et ceux non pertinents qui seront évités.

Makesense peut déplacer toutes les traces brutes depuis une machine donnée (par exemple une grille de calcul ou un front end IoT-lab) vers une autre (par exemple une station de travail locale ou un ordinateur portable).

Dans le cas d'une expérience sur IoT-lab, l'identifiant d'expérience qui est donné lors de la réservation d'une expérience permet d'obtenir l'adresse du dossier contenant les résultats sur le serveur front end. Ainsi Makesense récupère les traces dynamiquement en fonction de cet identifiant et sur l'ensemble des fichiers pertinents ce qui est plus efficace qu'une récupération manuelle de chaque fichier individuel.

Exploitation des résultats

L'exploitation de résultats est la phase permettant d'étudier de manière quantitative et qualitative une expérience en utilisant ses traces. Les étapes que cette phase comporte étant dissociées des contraintes des LLNs, tous les traitements peuvent utiliser des outils généraux et communs à de nombreux champs de recherche scientifique. Cette phase est complètement automatisable et déterministe, car elle applique des traitements fixes à des traces fournies en amont. L'objectif de cette phase consiste d'une part à décoder toutes les traces brutes pour en extraire les informations les plus pertinentes et d'autre part à fournir des données pour une interprétation quantitative et qualitative de l'expérience.

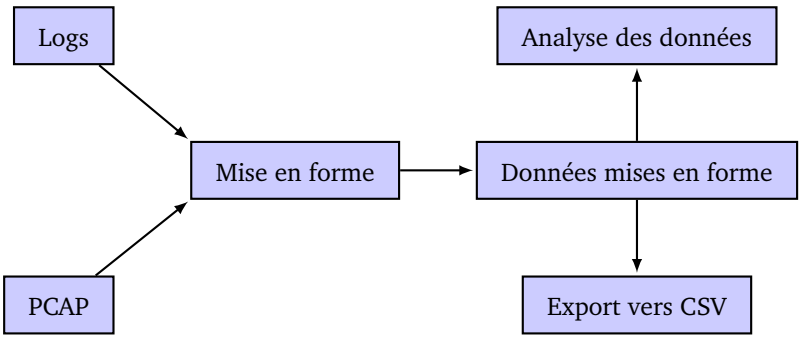


FIGURE 5.4 – Transformation des traces brutes vers des formats de données standards

La Figure 5.4 illustre l'approche choisie par Makesense. Le processus de transformation vise à aboutir à des données mises en forme qui seront exploitées directement lors de la phase d'analyse et qui peuvent être exportées au besoin vers un format standard pour être partagées.

Mise en forme de traces brutes

Les traces brutes sont de nature hétérogène, une première étape de la phase de traitement consiste à transformer ces résultats bruts vers des fichiers plus exploitables et homogènes. L'objectif de cette approche est de distinguer la phase de décodage d'une trace de son interprétation.

Dans le cas d'une expérience portant sur les LLNs, il s'agit par exemple de transformer les fichiers PCAP et les multiples journaux textes représentant les événements qui se sont produits par des transformations pour obtenir des traces plus exploitables.

Une conversion ad hoc aurait du mal à fonctionner pour un grand nombre de fichiers et de traitements différents. En outre, documenter les traitements effectués permet d'avoir une documentation sur les formats de documents obtenus afin d'éviter les traitements ad hoc. Enfin, transformer les données vers un format standard comme Comma Separated Values (CSV) permet de les importer dans de nombreux outils plus facilement notamment dans le cas de données tabulaires. Ainsi les mêmes traces peuvent être réutilisées pour des expériences voulant explorer différentes conclusions autour de données communes.

Makesense implémente ces transformations en utilisant des bibliothèques logicielles dédiées aux traitements de données tabulaires. L'objectif est d'appliquer à chaque élément d'une trace toutes les transformations nécessaires (renommage, normalisation, etc.) afin d'obtenir des tableaux et d'autres structures de données facilement exportables et manipulables.

Analyse des résultats

La phase d'analyse de résultats traite les données mises en forme afin de permettre leur interprétation et de fournir des résultats qualitatifs ou quantitatifs sur une expérience. Elle doit être facilitée par des itérations aussi courtes que possible pour tester de nouvelles hypothèses afin de diminuer le temps nécessaire pour confirmer ou infirmer une hypothèse sur une expérience.

Makesense met à profit des bibliothèques logicielles scientifiques afin d'effectuer ces traitements [127]. Les données mises en forme sont chargées dans des structures de données tabulaires où il est facile de les filtrer sur certains critères et de les agréger par différentes fonctions (une somme, un comptage, etc.). Le rendu de ces données peut être fait sous forme graphique avec des graphes qui illustrent les phénomènes observés.

Dans le cas d'expériences sur les LLNs, il est par exemple nécessaire d'exploiter les données disponibles dans les traces de trafic réseau en utilisant un dissecteur de paquets comme Wireshark [43]. Ces traces présentées dans le notebook sous la forme de tableau représentent pour chaque paquet les attributs trouvés par le dissecteur (source, destination, protocole, taille, etc.). Ainsi pour analyser la répartition de protocoles réseau, il suffit d'effectuer une requête similaire sémantiquement à une requête faite sur une base de données relationnelle classique [127]. Ce formalisme permet une grande expressivité sur les données recueillies et permet de s'abstraire des représentations concrètes des données dans les fichiers pour se concentrer sur ce qu'elles représentent. Ces données une fois trouvées peuvent être représentées par des visualisations et des graphes par des outils graphiques [87].

Un autre exemple consiste à avoir une représentation graphique de la topologie du réseau formée par les nœuds directement dans le notebook. Cela permet d'identifier rapidement les dysmétries et les nœuds importants pour la connexité du graphe. De plus, une fois cette représentation du DODAG RPL disponible, il est possible de calculer les plus courts chemins entre un nœud et la racine du DODAG afin d'obtenir la profondeur d'un nœud.

Ces fonctionnalités de groupement d'informations permettent d'exprimer de manière concise des traitements sur les données pour en extraire les informations utiles.

Conclusion

Ce chapitre montre qu'afin de permettre le partage et la reproductibilité d'une expérience, sa documentation et son automatisation jouent un rôle clé. Ce processus est assez systématique et dépasse largement le contexte des LLNs, ainsi des méthodologies et des outils communs peuvent émerger pour proposer des solutions à des problèmes communs.

Makesense est une réponse possible et se veut générique en utilisant des bibliothèques généralistes et répandues dans de multiples domaines de développement logiciel scientifique ou général. Makesense propose de documenter et d'automatiser une expérience pour LLN transcrite dans un notebook pour s'assurer de sa reproductibilité. La reproductibilité d'une expérience est une nécessité des projets de recherche. La réduction de la complexité de la mise en place d'une expérience permet d'accélérer une étude et d'augmenter son impact en facilitant sa vérification et son partage. À cette fin, Makesense est disponible sous licence Apache sur Github [9] et une démonstration du déroulement d'une expérience est également disponible [10]. Étant diffusé sous licence libre, il est difficile d'estimer son utilisation globale, cependant la version initiale disponible sur PyPI a été téléchargée plus de 1000 fois.

Makesense a été utilisé afin de réaliser les expériences décrites dans les chapitres précédents et pourrait être mis à profit afin d'avoir une solution permettant d'avoir une intégration complète d'une expérience dans le but de la diffuser.

Makesense ne gère pas pour le moment les erreurs à la volée qui peuvent subvenir lors d'une expérience. Or dans le cas de tests sur nœuds réels, ces erreurs peuvent être nombreuses par exemple dans le cas d'erreurs de flashage ou de panne d'un nœud. Il est alors nécessaire de redémarrer le nœud brutalement et cela ne garantit pas que le problème n'arrivera pas de nouveau.

Avoir une gestion plus fine et réactive des processus permettrait de prendre des décisions plus raisonnées dans le but d'avoir un environnement plus contrôlable. Cependant cette tâche est délicate, car les pannes et problèmes qui peuvent subvenir sont très hétérogènes et il est difficile de prendre des décisions automatiques dans ce genre de cas notamment dans le cas de testbeds mutualisées ou le contrôle de l'environnement et de ses interférences n'est pas facilement réalisable.

Une autre fonctionnalité pourrait être la gestion de points d'étapes ("checkpoint" en anglais) dans une expérience qui permettrait de sauvegarder l'intégralité du contexte d'une expérience à un instant donné. Cette fonctionnalité serait particulièrement utile pour effectuer des analyses post mortem d'un système tombé en panne.

Les méthodologies modernes de développement logiciel peuvent influencer les méthodologies expérimentales scientifiques notamment en informatique où de nombreux outils sont communs. La création de communautés autour des différents outils logiciels utilisés par la communauté scientifique facilite les échanges et la diffusion de nouvelles méthodes plus efficaces pour résoudre les problèmes communs. Makesense illustre comment intégrer efficacement ces technologies afin d'aider les expériences sur les LLNs à être plus reproductibles. Ce framework est mis à profit dans les expériences des chapitres 4 et 3.

Publications et invitations relatives à Makesense

— Rémy Leone, Jeremie Leguay, Paolo Medagliani, Claude Chaudet, et al. Makesense : Managing reproducible wsns experiments. *Fifth Workshop on Real-World Wireless Sensor Networks*, 2013.

9. https://github.com/sieben/makesense
10. https://travis-ci.org/sieben/makesense

— 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.
— Rémy Léone, Jérémie Leguay, Paolo Medagliani, and Claude Chaudet. Demo Abstract : Automating WSN experiments and simulations. In *EWSN*, 2015.
— Invitation au colloque R4 : Retour d'expéRiences sur la Recherche Reproductible. Le Studium - Centre International Universitaire pour la Recherche - Orléans 3 et 4 décembre 2015

Chapitre 6

Conclusion et perspectives

Passerelle avancée pour les LLNs

L'augmentation du nombre d'appareils connectés est continue et apporte de nouveaux défis pour l'intégration de ces sources de données dans les réseaux classiques existants. En raison de leur hétérogénéité, les nœuds des LLNs peuvent utiliser des technologies de communications différentes pour apporter une solution de connectivité efficace en fonction de leur environnement. La cohabitation de ces technologies semble inévitable car elles sont caractérisées par des compromis différents (bande passante, portée, consommation énergétique). Cette hétérogénéité des standards et des protocoles implique l'utilisation de solutions d'interconnexion qui sont le plus souvent bâties autour du protocole IPv6 et d'une passerelle se trouvant à la bordure de ces réseaux. Cette passerelle dispose de plusieurs radios lui permettant de communiquer physiquement mais aussi d'héberger des services supplémentaires par des fonctionnalités logicielles ou virtualisées en raison de ses ressources matérielles plus importantes. Ces services peuvent mitiger les problèmes d'interopérabilité car ils peuvent ajouter de manière logicielle du support de protocoles et de services tiers afin d'offrir de l'interopérabilité et de la découverte de services. Ainsi la passerelle peut devenir une plateforme sur laquelle de multiples services spécifiques aux LLNs peuvent être instanciés afin de notamment améliorer leurs performances ou leur fiabilité.

Cette thèse a présenté comment la passerelle qui est utilisée pour connecter un LLN à un réseau classique peut héberger des services pouvant optimiser de manière transparente l'utilisation des ressources d'un LLN. Le chapitre 2 a permis de montrer comment la passerelle du fait de sa position peut avoir accès à une grande quantité d'informations concernant le LLN. À partir de ces informations, plusieurs mécanismes permettant l'optimisation des ressources des LLNs ont étés proposés :

La Figure 6.1 illustre les contributions proposées dans cette thèse pouvant être déployées à la passerelle. Ces contributions sont détaillées ci-après.

Supervision

Le chapitre 3 a proposé une méthode de mesure implicite de l'utilisation de la radio en s'appuyant d'une part sur le trafic observé par le routeur de bordure et d'autre part sur la topologie de routage obtenue de manière opportuniste. En combinant ces observations avec une modélisation de la couche basse, il est possible d'avoir une estimation passive et transparente de l'utilisation de la radio pour tous les nœuds présents dans le LLN.

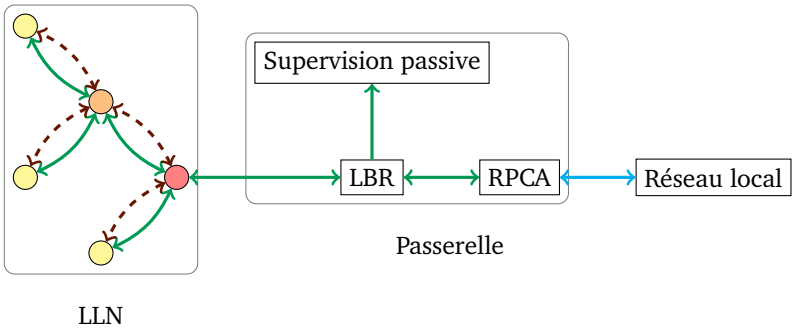


FIGURE 6.1 – Schéma de la passerelle proposée

La précision de cette estimation a été évaluée et a permis de conclure que la connaissance de la topologie améliorait grandement la précision de telles estimations. D'autre part, la validation expérimentale a permis de conclure que les phénomènes non prévisibles par la passerelle tels que les pertes de paquets ou les envois multiples d'un paquet par la couche MAC avaient une réelle influence. Ainsi la prévision à partir du trafic routé seul n'était pas suffisante pour avoir une vision de l'utilisation de la radio des nœuds d'un LLN sans le solliciter.

Mise en cache des réponses

Le chapitre 4 a proposé un mécanisme permettant d'optimiser l'utilisation des ressources d'un LLN en configurant un cache applicatif s'exécutant sur la passerelle afin de ne laisser passer qu'une quantité admissible de trafic vers le LLN. Ce mécanisme repose d'une part sur un RPC dont le but est de garder en cache les réponses aux requêtes visant un nœud du LLN et d'autre part sur une optimisation multi-objectifs capable de trouver des temps de vie en cache acceptables pour ces réponses. Les temps de vie pour chacune des réponses sont déterminés en prenant en compte à la fois les demandes des utilisateurs qui veulent avoir un contenu aussi récent que possible et d'autre part des nœuds du LLN qui veulent économiser leurs ressources en servant aussi peu de requêtes que possible.

La validation expérimentale a permis de confirmer que le mécanisme d'optimisation multi-objectifs produit le meilleur compromis entre des temps de vie acceptables et la satisfaction des utilisateurs. D'autre part, le RPCA peut être utilisé pour augmenter la durée de vie d'un LLN.

Reproductibilité des expériences

Le chapitre 5 a proposé Makesense : un framework permettant d'effectuer ces expériences de manière reproductible afin de garantir d'une part leur automatisation et d'autre part l'intégration de la documentation et du code dans un seul format cohérent. Makesense permet de garantir qu'une expérience est reproductible en transformant le notebook contenant l'expérience vers un programme qui une fois exécuté fournit le résultat d'une expérience. Makesense est diffusée sous licence libre [1] et les outils qu'il intègre peuvent être réutilisés dans des contextes variés.

1. https://github.com/sieben/makesense

Ouvertures

Les contributions proposées dans cette thèse peuvent être étendues dans plusieurs directions qui sont détaillées ici :

Mesure passive

En raison de leur hétérogénéité et de leurs ressources limitées, il est difficile de trouver des mécanismes de supervision généralistes fonctionnant sur tous les LLNs. L'ajout de mesures passives basées sur l'interprétation des flots réseau est un service qui peut être hébergée au niveau de la passerelle pour fournir des estimations minimales de consommation énergétique.

La précision de la supervision passive est liée à la fois à la précision de la modélisation de la couche MAC et d'autre part à l'utilisation de la radio que la passerelle peut inférer. Une piste d'amélioration de la supervision passive consisterait à l'appliquer dans des scénarios où les couches MAC et les conditions de trafic seraient beaucoup plus déterministes et donc plus facilement prévisibles.

Une autre piste d'amélioration se trouve au niveau de l'inférence de l'impact d'un paquet sur le LLN. La supervision passive proposée n'utilise que les entêtes source et destination pour comptabiliser une activité en se concentrant seulement sur son routage. Cependant des corrélations plus fines mettant en jeu une connaissance approfondie des protocoles et la corrélation de plusieurs paquets reçus (notamment les paquets de routage) permettraient d'avoir une connaissance plus fine des paquets qui ont été transmis dans le LLN mais que la passerelle ne peut observer. Cependant cette tâche est complexe, car les implémentations ne respectent pas toujours les spécifications ainsi l'observation d'un paquet peut avoir différentes interprétations selon l'implémentation mise en jeu.

Reverse Proxy Cache Adaptatif

Contrôler et mesurer une métrique est une nécessité pour ensuite agir et prendre des décisions sur le fonctionnement du réseau. La régulation des requêtes entrantes sur un LLN est un service qui peut être instancié au niveau de la passerelle qui permet d'une part de réguler la quantité de trafic applicatif mais également d'offrir un service d'accélération du traitement des requêtes.

Le rôle d'un RPCA consiste à adapter les temps de vie des réponses des requêtes applicatives afin de satisfaire au mieux à la fois les nœuds du LLN et les utilisateurs. L'heuristique employée dans cette thèse est basée sur des algorithmes génétiques, cependant il est possible que des heuristiques plus efficaces puissent être utilisées pour trouver les temps de vie. Il est à noter que les heuristiques employées pour trouver ces temps de vie peuvent être très complexes notamment dans le cas où la modélisation de la durée de vie met en jeu un grand nombre d'éléments. Un comparatif des différentes heuristiques possibles pourrait permettre de trouver en fonction de la modélisation du problème l'heuristique la plus adaptée. Cependant cette tâche est complexe en raison de la variété des heuristiques disponibles dans la littérature et de la difficulté de les configurer efficacement.

Une autre contribution possible serait d'avoir une modélisation de la satisfaction des utilisateurs tenant compte de la popularité d'une URI plutôt que des temps de vie en cache qui lui sont appliqués. Cette approche pourrait permettre d'avoir des gestions beaucoup plus dynamiques de la qualité de service ressentie par les utilisateurs. Cependant mettre une "valeur" sur une URI ouvre la voie à des modélisations complexes pouvant par exemple se ramener à des problèmes de sac-à-dos multi-objectifs qui peuvent être difficiles à résoudre efficacement dans un cas général avec un grand nombre d'URI.

Reproductibilité en simulations et expérimentations

L'une des contributions les plus souhaitables pour Makesense serait de disposer d'une gestion des "snapshot" permettant d'avoir un instantané d'une simulation ou d'une expérience en cours. Cependant bien que ce type de fonctionnalité soit possible dans le cas de simulateur ou d'émulateur, il est difficile à obtenir sur des nœuds réels car l'état exact d'un système embarqué est difficile à obtenir dans le cas général.

D'autre part, une expérience sur les LLNs met également en jeu de nombreux processus concurrents (agrégation des données, injection de trafic, etc.) or la gestion de ces processus et de leur cycle de vie est parfois complexe rendant cette abstraction délicate. Une autre piste de contribution serait une gestion de haut niveau du cycle de vie de ces processus afin de garantir un niveau de fiabilité suffisant au cours de l'expérience.

Usages potentiels

Les contributions présentées dans cette thèse sont applicables dans plusieurs champs à la fois en recherche et en développement industriel :

Recherche reproductible

Les contributions présentées par Makesense peuvent être mises à profit pour diffuser plus largement des expériences mettant en jeu des LLN. Une première piste d'application serait la constitution de supports pédagogiques interactifs pour permettre de diffuser des expériences mettant en jeu des LLN simulés, d'observer leur comportement puis d'effectuer les mêmes expériences sur des nœuds réels. L'utilisation de support permettrait de raccourcir le temps nécessaire pour exécuter une expérience sur différents supports : nœuds émulés pour démarrer, puis transition vers des nœuds physiques pour confirmer des premiers résultats obtenus et enfin exécuter une expérience à large échelle.

D'autre part, Makesense peut également être utilisé lors de recherches sur les LLN pour gérer efficacement à la fois l'obtention des données, leur visualisation et la documentation d'un protocole expérimental. Documenter des expériences ou des démonstrations avec Makesense permettrait de pouvoir les ré-exécuter plus facilement et de garantir un suivi de leur fiabilité au cours du temps grâce à l'intégration continue.

Supervision passive

La mesure implicite présentée dans cette thèse peut être utilisée par des solutions de supervision intégrée à destination des LLN. Par exemple, cette solution de supervision permettrait de voir dans son interface de contrôle des estimations de trafic réseau et de la consommation énergétique estimée. L'avantage de la supervision passive est de garantir que quelque soit la configuration des nœuds, une estimation minimale de l'utilisation de la radio soit toujours disponible.

RPC dans un contexte industriel

Le RPCA présenté dans cette thèse peut être utilisé sur un reverse-proxy situé devant de multiples serveurs applicatifs fonctionnant dans un LLN. Les RPC disposent le plus souvent d'un système de modules afin d'ajouter des fonctionnalités en fonction des déploiements. Le RPCA présenté ici pourrait être diffusé sous la forme de modules pour les RPC les plus utilisés dans l'industrie afin que les temps de vie puissent être gérés dynamiquement et permettrait ainsi de réguler la quantité de requêtes gérée par le LLN en fonction de son état. Le RPCA présenté dans cette thèse a d'ailleurs été utilisé

lors du projet européen Calipso afin de faire la démonstration de l'utilité des LLN basés sur IP pour construire l'Internet des objets à venir dans les années futures.

Annexe A

Collaborations extérieures

A scalable and self-configuring architecture for service discovery in the internet of things

Les déploiements visés par l'IoT mettent en jeu des milliards d'appareils, ainsi ces appareils rendent de nouvelles formes d'interactions entre les personnes et les objets possibles.

Afin de permettre des applications robustes et faciles d'utilisation sur ces objets, il est nécessaire de disposer de mécanismes qui minimisent (ou annule) le besoin d'intervention humaine extérieure pour leur configuration et leur maintenance. Ces mécanismes doivent pouvoir gérer un grand nombre d'objets qui sera en forte croissance au cours des prochaines années.

Cette contribution propose un mécanisme d'auto-configuration basée sur une architecture pair-à-pair, visant de grands réseaux d'objets et permettant d'obtenir un service de découverte de ressource automatisée ne nécessitant aucune intervention humaine pour leur configuration. En particulier, cette contribution se concentre sur les mécanismes de découverte de services locaux et globaux en montrant comment l'architecture proposée permet une interaction tout en préservant une indépendance opérationnelle.

L'efficacité de l'architecture proposée a été confirmée par des résultats expérimentaux obtenus sur déploiement réel.

Publication

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.

Bounding Degrees on RPL

RPL est le protocole de routage standardisé par l'IETF optimisé pour les LLNs.

La stabilité de RPL peut souffrir de pertes de paquets et causer une instabilité des routes. La plupart des solutions dédiées à ce problème se concentrent sur l'amélioration des métriques utilisées pour construire les routes. Ces métriques sont le plus souvent basées sur une évaluation de la qualité du lien radio.

BD-RPL (Bounded Degree RPL) adopte une nouvelle approche en ajoute une contrainte supplémentaire sur le nombre maximal d'enfants qu'un routeur RPL peut accepter lors de la construction des routes. Cette méthode utilise les paquets de signalisation utilisés par défaut par RPL et ne nécessite pas une nouvelle signalisation.

BD-RPL est évalué sur simulateur et implémenté sur des nœuds réels qui ont montré une amélioration de la fiabilité des transmissions de 10%, une réduction de la consommation énergétique de 50% et une amélioration de 60% du délai.

Publication

Fadwa Boubekeur, Lélia Blin, Remy Leone, and Paolo Medagliani. Bounding degrees on rpl. In *Proceedings of the 11th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, pages 123–130. ACM, 2015.

Acronymes

h Cache Hit ratio
m Cache Miss ratio
6LBR 6LoWPAN Border Router
6LoWPAN IPv6 over Low power Wireless Personal Area Networks
ACK Acknowledgement message
AODV Ad-hoc On-demand Distance Vector
API Application Programming Interface
ARCEP Autorité de régulation des communications électroniques et des postes
ARP Address Resolution Protocol
BAN Body-Area Network
BGP Border Gateway Protocol
BLE Bluetooth Low-Energy
BSS Basic Service Set
CBOR Concise Binary Object Representation
CCA Channel Check Assessment
CCN Content Centric Network
CoAP Constrained Application Protocol
CON Confirmable message
CoRE Constrained RESTful environment
CPL Courant Porteur en Ligne
CPU Central Processing Unit
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
DHCP Dynamic Host Configuration Protocol

DIO DODAG Information Object
DIS DODAG Informational Solicitation
DNS Domain Name System
DODAG Destination-Oriented DAG
DSL Domain Specific Language
DTLS Datagram Transport Layer Security
ETag Entity Tag
ETX Expected Transmission Count
EUI Extended Unique Identifier
EWMA Exponentially Weighted Moving Average
FFD Full Function Device
HCP HTTP-CoAP proxy
HTTP HyperText Transfer Protocol
ICMPv6 Internet Control Message Protocol v6
ICN Information-Centric Networking
IEEE Institute of Electrical and Electronics Engineers
IETF Internet Engineering Task Force
IGP Interior Gateway Protocol
IHM Interaction Homme Machine
IID Interface ID
IoT Internet of Things
IP Internet Protocol
ISM Bande industrielle, scientifique et médicale
JSON Javascript Serial Object Notation
KP Knapsack Problem
LAN Local Area Network
LBR LoWPAN Border Router
LLN Low-Power and Lossy Network
LoWPAN Low-Power Wireless Personal Area Network
LPWAN Low-Power Wide Area Network
LRWPAN Low Rate Wireless Personal Area Network
M2M Machine to Machine
MAC Media Access Control
MITM Man-in-the-middle
MP2P Multi-point to point
MQTT Message Queuing Telemetry Transport
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
NTP Network Time Protocol
OEDL OMF Experiment Description Language
OLSR Optimized Link State Routing protocol
OMF Orbit Management Framework
OSPF Open Shortest Path First
OTA Over The Air
P2MP Point to Multi-point
P2P Point to Point
PAN Personnal Area Network
PCAP Packet CAPture
PDF Portable Document Format
PDR Packet Delivery Ratio
PLSDU Physical Layer Service Data Unit
PoE Power over Ethernet
PSM Power Saving Mode
QoS Quality of Service
REST REpresentational State Transfer
RFD Reduced Function Device
RMAB Restless Multi-Armed Bandit
ROLL Routing Over Low power and Lossy Networks
RPCA Reverse Proxy Cache Adaptatif
RPC Reverse Proxy Cache
RPL Routing Protocol Layer
RST Reset message
SCADA Supervisory Control and Data Acquisition
SICS Swedish Institute of Computer Science
SOA Service Oriented Architecturei
SQL Structured Query Language
SSH Secure Shell
TCP Transport Control Protocol
TDD Test Driven Development
TDMA Time Division Multiple Access
Tee Traffic Energy Estimator
TSCH Time-Slotted Channel Hopping

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] Caching best practices and max-age gotchas. `https://jakearchibald.com/2016/caching-best-practices`. Accessed : 2016-04-30.

[2] The lightweight on-demand ad hoc distance-vector routing protocol - next generation (loadng). Technical report.

[3] 6tisch. IPv6 over the TSCH mode of IEEE 802.15.4e. `http://datatracker.ietf.org/wg/6tisch/`.

[4] Ameer Ahmed Abbasi and Mohamed Younis. A survey on clustering algorithms for wireless sensor networks. *Computer communications*, 30(14) :2826–2841, 2007.

[5] Nazim Abdeddaim and Fabrice Theoleyre. Implementation of a wsnet module to simulate the ieee 802.15. 4 beacon-enabled mode in multihop topologies. 2011.

[6] 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.

[7] ZigBee Alliance. Zigbee specification, 2006.

[8] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. *Web services*. Springer, 2004.

[9] Giuseppe Anastasi, Marco Conti, Enrico Gregori, and Andrea Passarella. 802.11 power-saving mode for mobile computing in wi-fi hotspots : limitations, enhancements and open issues. *Wireless Networks*, 14(6) :745–768, 2008.

[10] D Antolin, N Medrano, and B Calvo. Analysis of the operating life for battery-operated wireless sensor nodes. In *Industrial Electronics Society, IECON 2013-39th Annual Conference of the IEEE*, pages 3883–3886. IEEE, 2013.

[11] Carles Anton-Haro and Mischa Dohler. *Machine-to-machine (M2M) Communications : Architecture, Performance and Applications*. Elsevier, 2014.

[12] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things : A survey. *Computer networks*, 54(15) :2787–2805, 2010.

[13] Jordan Augé, Thierry Parmentelat, Nicolas Turro, Sandrine Avakian, Loïc Baron, Mohamed Amine Larabi, Mohammed Yasin Rahman, Timur Friedman, and Serge Fdida. Tools to foster a global federation of testbeds. *Computer Networks*, 63 :205–220, 2014.

[14] 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.

[15] 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.

[16] Gaston Bachelard. nouvel esprit scientifique [le]. 1975.

[17] Debasis Bandyopadhyay and Jaydip Sen. Internet of things : Applications and challenges in technology and standardization. *Wireless Personal Communications*, 58(1) :49–69, 2011.

[18] Loc Baron, Jordan Augé, Timur Friedman, and Serge Fdida. Towards an integrated portal for networking testbed federation, an open platform approach. In *FIRE Engineering workshop, Ghent, Belgium*, 2012.

[19] 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.

[20] Kent Beck. *Test-driven development : by example*. Addison-Wesley Professional, 2003.

[21] DS Berger, P Gland, S Singla, and F Ciucu. Exact analysis of ttl cache networks : The case of caching policies driven by stopping times. arxiv preprint. *arXiv*, 1402, 2014.

[22] Emerging Technologies Blog. Powered by jupyter : A survey of the project ecosystem. `http://blog.ibmjstart.net/2016/03/21/powered-by-jupyter`.

[23] Georg Brandl. Sphinx documentation. *URL http ://sphinx-doc. org/sphinx. pdf*, 2010.

[24] 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.

[25] Nevil Brownlee and KC Claffy. Understanding internet traffic streams : dragonflies and tortoises. *Communications Magazine, IEEE*, 40(10) :110–117, 2002.

[26] 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.

[27] Bernhard Buchli, Daniel Aschwanden, and Jan Beutel. Battery state-of-charge approximation for energy harvesting embedded systems. In *Wireless Sensor Networks*, pages 179–196. Springer, 2013.

[28] 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.

[29] Andrea Caragliu, Chiara Del Bo, and Peter Nijkamp. Smart cities in europe. *Journal of urban technology*, 18(2) :65–82, 2011.

[30] Edward Chan and Song Han. Energy efficient residual energy monitoring in wireless sensor networks. *International Journal of Distributed Sensor Networks*, 5(6), 2009.

[31] Bor-rong Chen, Geoffrey Peterson, Geoff Mainland, and Matt Welsh. Livenet : Using passive monitoring to reconstruct sensor network dynamics. In *IEEE/ACM DCOSS*, 2008.

[32] Yunxia Chen and Qing Zhao. On the lifetime of wireless sensor networks. *Communications Letters, IEEE*, 9(11) :976–978, 2005.

[33] Nakjung Choi, Kyle Guan, Daniel C Kilper, and Gary Atkinson. In-network caching effect on optimal energy consumption in content-centric networking. In *Communications (ICC), 2012 IEEE International Conference on*, pages 2889–2894. IEEE, 2012.

[34] 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.

[35] B. Claise, B. Trammell, and P. Aitken. Specification of the ip flow information export (ipfix) protocol for the exchange of flow information. STD 77, RFC Editor, September 2013. `http://www.rfc-editor.org/rfc/rfc7011.txt`.

[36] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626 (Experimental), October 2003.

[37] 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.

[38] Lorenzo Colitti, Steinar H Gunderson, Erik Kline, and Tiziana Refice. Evaluating ipv6 adoption in the internet. In *Passive and Active Measurement*, pages 141–150. Springer, 2010.

[39] W. Colitti, K. Steenhaut, and N. De Caro. Integrating wireless sensor networks with the web. *Extending the Internet to Low powerand Lossy Networks (IP+ SN 2011)*, 2011.

[40] 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.

[41] Matteo Collina, Giovanni Emanuele Corazza, and Alessandro Vanelli-Coralli. Introducing the qest broker : Scaling the iot by bridging mqtt and rest. In *Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on*, pages 36–41. IEEE, 2012.

[42] R. Coltun, D. Ferguson, J. Moy, and A. Lindem. OSPF for IPv6. RFC 5340 (Proposed Standard), July 2008. Updated by RFCs 6845, 6860, 7503.

[43] Gerald Combs. Tshark dump and analyze network traffic. `https://www.wireshark.org/docs/man-pages/tshark.html`.

[44] Alex Conta. Generic packet tunneling in ipv6 specification. 1998.

[45] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. Social coding in github : transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1277–1286. ACM, 2012.

[46] Bart De Schutter and Bart De Moor. Optimal traffic light control for a single intersection. *European Journal of Control*, 4(3) :260–276, 1998.

[47] 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.

[48] Kalyanmoy Deb and Debayan Deb. Analysing mutation schemes for real-parameter genetic algorithms. *International Journal of Artificial Intelligence and Soft Computing*, 4(1) :1–28, 2014.

[49] Stephen E Deering. Internet protocol, version 6 (ipv6) specification. 1998.

[50] Manjunath Doddavenkatappa, Mun Choon Chan, and Akkihebbal L Ananda. Indriya : A low-cost, 3d wireless sensor network testbed. In *Testbeds and Research Infrastructure. Development of Networks and Communities*, pages 302–316. Springer, 2011.

[51] Manjunath Doddavenkatappa, Mun Choon Chan, and Akkihebbal L Ananda. Indriya : A low-cost, 3d wireless sensor network testbed. In *Testbeds and Research Infrastructure. Development of Networks and Communities*, pages 302–316. Springer, 2012.

[52] Enrique J Duarte-Melo and Mingyan Liu. Analysis of energy consumption and lifetime of heterogeneous wireless sensor networks. In *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE*, volume 1, pages 21–25. IEEE, 2002.

[53] A. Dunkels, B. Gronvall, and T. Voigt. Contiki-a lightweight and flexible operating system for tiny networked sensors. pages 455–462, 2004.

[54] Adam Dunkels. The ContikiMAC Radio Duty Cycling Protocol. Technical Report T2011 :13, Swedish Institute of Computer Science, December 2011.

[55] Adam Dunkels, Joakim Eriksson, Niclas Finne, and Nicolas Tsiftes. Powertrace : Network-level power profiling for low-power wireless networks. 2011.

[56] Simon Duquennoy, Niklas Wirström, 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.

[57] Paul M Duvall, Steve Matyas, and Andrew Glover. *Continuous integration : improving software quality and reducing risk*. Pearson Education, 2007.

[58] David Egan. The emergence of zigbee in building automation and industrial control. *Computing & Control Engineering Journal*, 16(2) :14–19, 2005.

[59] Gregory A Ehlers, Robert D Howerton, and Gary E Speegle. Engery management and building automation system, November 5 1996. US Patent 5,572,438.

[60] 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.

[61] Hossein Falaki, Ratul Mahajan, Srikanth Kandula, Dimitrios Lymberopoulos, Ramesh Govindan, and Deborah Estrin. Diversity in smartphone usage. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 179–194. ACM, 2010.

[62] Sébastien Faye, Claude Chaudet, and Isabelle Demeure. A distributed algorithm for adaptive traffic lights control. In *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pages 1572–1577. IEEE, 2012.

[63] Stuart I Feldman. Make—a program for maintaining computer programs. *Software : Practice and experience*, 9(4) :255–265, 1979.

[64] Konstantinos P Ferentinos and Theodore A Tsiligiridis. Adaptive design optimization of wireless sensor networks using genetic algorithms. *Computer Networks*, 51(4) :1031–1051, 2007.

[65] Roy Fielding and J Reschke. Rfc 7234-hypertext transfer protocol (http/1.1) : Caching. *URl : http ://tools. ietf. org/html/rfc7234 (v isited on 02/19/2015)*.

[66] Arlene Fink. *Conducting Research Literature Reviews : From the Internet to Paper : From the Internet to Paper*. Sage Publications, 2013.

[67] 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.

[68] Eric Fleury, Nathalie Mitton, Thomas Noel, Cédric Adjih, Valeria Loscri, Anna Maria Vegni, Riccardo Petrolo, Valeria Loscri, Gianluca Aloi, et al. Fit iot-lab : The largest iot open experimental testbed. *ERCIM News*, (101) :14, 2015.

[69] Pietro Gonizzi, Paolo Medagliani, Giorgio Ferrari, and Jeremie Leguay. Rawmac : A routing aware wave-based mac protocol for wsns. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2014 IEEE 10th International Conference on*, pages 205–212. IEEE, 2014.

[70] David Gourley and Brian Totty. *HTTP : the definitive guide*. " O'Reilly Media, Inc.", 2002.

[71] Theodore W Gray and Jerry Glynn. *Exploring Mathematics with Mathematica : Dialogs Concerning Computers and Mathematics*. Addison-Wesley Longman Publishing Co., Inc., 1991.

[72] 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.

[73] 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.

[74] 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.

[75] Liu Ying Liu Jing Zheng Haiyan. The application of cacti in the campus network traffic monitoring [j]. *Computer & Telecommunication*, 4 :004, 2008.

[76] Vlado Handziski, Andreas Köpke, Andreas Willig, and Adam Wolisz. Twist : a scalable and reconfigurable testbed for wireless indoor experiments with sensor networks. In *Proceedings of the 2nd international workshop on Multi-hop ad hoc networks : from theory to reality*, pages 63–70. ACM, 2006.

[77] Peter Harrington. *Machine learning in action*. Manning, 2012.

[78] Jane K Hart and Kirk Martinez. Environmental sensor networks : A revolution in the earth system science ? *Earth-Science Reviews*, 78(3) :177–191, 2006.

[79] K. Hartke. Observing Resources in the Constrained Application Protocol (CoAP). RFC 7641 (Proposed Standard), September 2015.

[80] Lorin Hochstein. *Ansible : Up and Running*. " O'Reilly Media, Inc.", 2014.

[81] Robert G Hollands. Will the real smart city please stand up ? intelligent, progressive or entrepreneurial ? *City*, 12(3) :303–320, 2008.

[82] 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.

[83] Yih-Chun Hu, David B Johnson, and Adrian Perrig. Sead : Secure efficient distance vector routing for mobile wireless ad hoc networks. *Ad hoc networks*, 1(1) :175–192, 2003.

[84] J. Hui and JP. Vasseur. The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams. RFC 6553 (Proposed Standard), March 2012.

[85] J. Hui, JP. Vasseur, D. Culler, and V. Manral. An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL). RFC 6554 (Proposed Standard), March 2012.

[86] Urs Hunkeler, Hong Linh Truong, and Andy Stanford-Clark. Mqtt-s—a publish/subscribe protocol for wireless sensor networks. In *Communication systems software and middleware and workshops, 2008. comsware 2008. 3rd international conference on*, pages 791–798. IEEE, 2008.

[87] J. D. Hunter. Matplotlib : A 2d graphics environment. *Computing In Science & Engineering*, 9(3) :90–95, 2007.

[88] Sajid Hussain, Abdul W Matin, and Obidul Islam. Genetic algorithm for energy efficient clusters in wireless sensor networks. In *null*, pages 147–154. IEEE, 2007.

[89] Miloš Jevtić, Nikola Zogović, and Goran Dimić. Evaluation of wireless sensor network simulators. In *Proceedings of the 17th Telecommunications Forum (TELFOR 2009), Belgrade, Serbia*, pages 1303–1306. Citeseer, 2009.

[90] 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.

[91] Vasileios Karagiannis, Periklis Chatzimisios, Francisco Vazquez-Gallego, and Jesus Alonso-Zarate. A survey on application layer protocols for the internet of things. *Transaction on IoT and Cloud Computing*, 3(1) :11–17, 2015.

[92] 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.

[93] Fotis Kerasiotis, Aggeliki Prayati, Christos Antonopoulos, Christos Koulamas, and George Papadopoulos. Battery lifetime prediction model for a wsn platform. In *Sensor Technologies and Applications (SENSORCOMM), 2010 Fourth International Conference on*, pages 525–530. IEEE, 2010.

[94] Branko Kerkez, Steven D Glaser, Roger C Bales, and Matthew W Meadows. Design and performance of a wireless sensor network for catchment-scale snow and soil moisture measurements. *Water Resources Research*, 48(9), 2012.

[95] Kavi K Khedo, Rajiv Perseedoss, Avinash Mungur, et al. A wireless sensor network air pollution monitoring system. *arXiv preprint arXiv :1005.1737*, 2010.

[96] Hee-Woong Kim, Hock Chuan Chan, and Sumeet Gupta. Value-based adoption of mobile internet : an empirical investigation. *Decision Support Systems*, 43(1) :111–126, 2007.

[97] 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.

[98] Ross D King, Jem Rowland, Stephen G Oliver, Michael Young, Wayne Aubrey, Emma Byrne, Maria Liakata, Magdalena Markham, Pinar Pir, Larisa N Soldatova, et al. The automation of science. *Science*, 324(5923) :85–89, 2009.

[99] 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.

[100] Hermann Kopetz. *Real-time systems : design principles for distributed embedded applications*. Springer Science & Business Media, 2011.

[101] 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.

[102] 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.

[103] 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.

[104] Abdelkader Lahmadi, Alexandre Boeglin, and Olivier Festor. Efficient distributed monitoring in 6lowpan networks. In *CNSM*, Zurich, Switzerland, 2013.

[105] Chih-Chung Lai, Chuan-Kang Ting, and Ren-Song Ko. An effective genetic algorithm to improve wireless sensor network lifetime for large-scale surveillance applications. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 3531–3538. IEEE, 2007.

[106] Koen Langendoen. Medium access control in wireless sensor networks. *Medium access control in wireless networks*, 2 :535–560, 2008.

[107] Jung Hoon Lee, Robert Phaal, and Sang-Ho Lee. An integrated service-device-technology roadmap for smart city development. *Technological Forecasting and Social Change*, 80(2) :286–306, 2013.

[108] Kyunghan Lee, Joohyun Lee, Yung Yi, Injong Rhee, and Song Chong. Mobile data offloading : how much can wifi deliver ? In *Proceedings of the 6th International COnference*, page 26. ACM, 2010.

[109] Yongwon Lee and Scott H Clearwater. Tools for automating experiment design : A machine learning approach. In *Tools with Artificial Intelligence, 1992. TAI'92, Proceedings., Fourth International Conference on*, pages 324–331. IEEE, 1992.

[110] Tomas Lennvall, Stefan Svensson, and Fredrik Hekland. A comparison of wirelesshart and zigbee for industrial applications. In *IEEE International Workshop on Factory Communication Systems*, volume 2008, pages 85–88, 2008.

[111] Rémy Leone, Paolo Medagliani, and Jérémie Leguay. Optimizing qos in wireless sensors networks using a caching platform. In *Sensornets 2013*, page 56, 2013.

[112] P. Levis. Overview of Existing Routing Protocols for Low Power and Lossy Networks. Internet-Draft draft-ietf-roll-protocols-survey-07, Internet Engineering Task Force. Work in progress.

[113] P. Levis, T. Clausen, J. Hui, O. Gnawali, and K. J. The Trickle Algorithm. IETF RFC 6206, 2011.

[114] Jun Li, Bin Liu, and Hao Wu. Energy-efficient in-network caching for content-centric networking. *Communications Letters, IEEE*, 17(4) :797–800, 2013.

[115] Slawek Ligus. *Effective Monitoring and Alerting*. " O'Reilly Media, Inc.", 2012.

[116] Changlei Liu and Guohong Cao. Distributed monitoring and aggregation in wireless sensor networks. In *IEEE INFOCOM*, San Diego, CA, USA, 2010.

[117] Keqin Liu and Qing Zhao. Intrusion detection in resource-constrained cyber networks : A restless multi-armed bandit approach. *submitted to IEEE/ACM Transactions on Networking. Avialable at http ://arxiv. org/abs*, 1112.

[118] Jaime Llorca, Antonia M Tulino, Ke Guan, Jairo Esteban, Matteo Varvello, Nakjung Choi, and Daniel Kilper. Dynamic in-network caching for energy efficient content delivery. In *INFOCOM, 2013 Proceedings IEEE*, pages 245–249. IEEE, 2013.

[119] Jon Loeliger and Matthew McCullough. *Version Control with Git : Powerful tools and techniques for collaborative software development*. " O'Reilly Media, Inc.", 2012.

[120] David G Loomis and Lester D Taylor. *Forecasting the Internet : understanding the explosive growth of data communications*, volume 39. Springer Science & Business Media, 2012.

[121] Sean Luke. *Essentials of Metaheuristics*. Lulu, second edition, 2013. Available for free at http ://cs.gmu.edu/∼sean/book/metaheuristics/.

[122] 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.

[123] Arun Manivannan. *Scala Data Analysis Cookbook*. Packt Publishing, 2015.

[124] Jerry Martocci, Pieter Mil, Nicolas Riou, and Wouter Vermeylen. Building automation routing requirements in low-power and lossy networks. 2010.

[125] E Michael Maximilien and Laurie Williams. Assessing test-driven development at ibm. In *Software Engineering, 2003. Proceedings. 25th International Conference on*, pages 564–569. IEEE, 2003.

[126] Steve McCanne, Craig Leres, and Van Jacobson. Libpcap, 1989.

[127] Wes McKinney. *Python for data analysis : Data wrangling with Pandas, NumPy, and IPython*. " O'Reilly Media, Inc.", 2012.

[128] 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.

[129] Mathieu Michel and Quoitin Bruno. Analyse des raisons de l'efficacité de contikimac. In *ALGOTEL 2014–16èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*, pages 1–4, 2014.

[130] Scott Michel, Khoi Nguyen, Adam Rosenstein, Lixia Zhang, Sally Floyd, and Van Jacobson. Adaptive web caching : towards a new global caching architecture. *Computer Networks and ISDN systems*, 30(22) :2169–2177, 1998.

[131] 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.

[132] Javier Moreno Molina, Jan Haase, and Christoph Grimm. Energy consumption estimation and profiling in wireless sensor networks. In *Architecture of Computing Systems (ARCS), 2010 23rd International Conference on*, pages 1–6. VDE, 2010.

[133] 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.

[134] A. Morton. Active and passive metrics and methods (with hybrid types in-between). RFC 7799, RFC Editor, May 2016.

[135] San Murugesan. Harnessing green it : Principles and practices. *IT professional*, 10(1) :24–33, 2008.

[136] Emmanuel Nataf and Olivier Festor. Online estimation of battery lifetime for wireless sensors network. *arXiv preprint arXiv :1209.2234*, 2012.

[137] 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.

[138] Fredrik Österlind, Adam Dunkels, Joakim Eriksson, Niclas Finne, and Thiemo Voigt. Cross-level sensor network simulation with cooja. In *LCN*, 2006.

[139] Bryan O'Sullivan. Making sense of revision-control systems. *Communications of the ACM*, 52(9) :56–62, 2009.

[140] 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.

[141] 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.

[142] Roger D Peng. Reproducible research in computational science. *Science (New York, Ny)*, 334(6060) :1226, 2011.

[143] Fernando Pérez and Brian E. Granger. IPython : a system for interactive scientific computing. *Computing in Science and Engineering*, 9(3) :21–29, May 2007.

[144] Fernando Perez, Brian E Granger, and CPSL Obispo. An open source framework for interactive, collaborative and reproducible scientific computing and education, 2013.

[145] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 (Experimental), July 2003.

[146] Peter Phaal, Sonia Panchen, and Neil McKee. Inmon corporation's sflow : A method for monitoring traffic in switched and routed networks. Technical report, 2001.

[147] Don T Phillips and Alberto Garcia-Diaz. *Fundamentals of network analysis*. Prentice Hall, 1981.

[148] Joern Ploennigs, Uwe Ryssel, and Klaus Kabitzsch. Performance analysis of the enocean wireless sensor network protocol. In *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*, pages 1–9. IEEE, 2010.

[149] 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.

[150] Karl Raimund Popper. *The Logic of Scientific Discovery*. Routledge, 2002. 1st English Edition :1959.

[151] Hart E Posen and Daniel A Levinthal. Chasing a moving target : Exploitation and exploration in dynamic environments. *Management Science*, 58(3) :587–601, 2012.

[152] Jupyter project. The notebook file format. `https://nbformat.readthedocs.org/en/latest/format_description.html`.

[153] Daniele Puccinelli and Martin Haenggi. Multipath fading in wireless sensor networks : measurements and interpretation. In *Proceedings of the 2006 international conference on Wireless communications and mobile computing*, pages 1039–1044. ACM, 2006.

[154] Guy Pujolle. *Les réseaux : Edition 2014*. Editions Eyrolles, 2014.

[155] Thierry Rakotoarivelo, Maximilian Ott, Guillaume Jourjon, and Ivan Seskar. Omf : a control and management framework for networking testbeds. *ACM SIGOPS Operating Systems Review*, 43(4) :54–59, 2010.

[156] 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.

[157] ERC Rec. 70–03 : Relating to the use of short range devices (srd), 2001.

[158] Will Reese. Nginx : the high-performance web server and reverse proxy. *Linux Journal*, 2008(173) :2, 2008.

[159] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard), January 2006. Updated by RFCs 6286, 6608, 6793, 7606, 7607.

[160] Leonard Richardson and Sam Ruby. *RESTful web services*. " O'Reilly Media, Inc.", 2008.

[161] 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.

[162] Z. Shelby. Constrained RESTful Environments (CoRE) Link Format. RFC 6690 (Proposed Standard), August 2012.

[163] Z. Shelby, K. Hartke, and C. Bormann. The Constrained Application Protocol (CoAP). RFC 7252 (Proposed Standard), June 2014.

[164] Zach Shelby and Carsten Bormann. *6LoWPAN : The wireless embedded Internet*, volume 43. John Wiley & Sons, 2011.

[165] Helen Shen et al. Interactive notebooks : Sharing the code. *Nature*, 515(7525) :151–152, 2014.

[166] Silvair. Bluetooth : A technology in transition. `https://blog.silvair.com/2015/11/26/bluetooth-a-technology-in-transition/`, 2015.

[167] Silvair. Riding the z-wave. https://blog.silvair.com/2015/10/15/wireless-protocols-showdown-riding-the-z-wave/, 2015.

[168] Silvair. Threading the way through a connected home. https://blog.silvair.com/2015/11/12/wireless-protocols-showdown-threading-the-way-through-a-connected-home 2015.

[169] Silvair. Wireless protocols showdown : Why not wi-fi ? https://blog.silvair.com/2015/10/01/wireless-protocols-showdown-3/, 2015.

[170] Silvair. Wireless protocols showdown : Zigbee – is the sting still sharp ? https://blog.silvair.com/2015/10/27/zigbee-is-the-sting-still-sharp/, 2015.

[171] Ahmed Sobeih, Jennifer C Hou, Lu-Chuan Kung, Ning Li, Honghai Zhang, Wei-Peng Chen, Hung-Ying Tyan, and Hyuk Lim. J-sim : a simulation and emulation environment for wireless sensor networks. *Wireless Communications, IEEE*, 13(4) :104–119, 2006.

[172] 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.

[173] Kenneth Sörensen. Metaheuristics—the metaphor exposed. *International Transactions in Operational Research*, 22(1) :3–18, 2015.

[174] AS Stanford-Clark and Hong Linh Truong. Mqtt for sensor networks (mqtt-s) protocol specification. *International Business Machines Corporation version*, 1, 2008.

[175] Luka Stanisic, Arnaud Legrand, and Vincent Danjean. An effective git and org-mode based workflow for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1) :61–70, 2015.

[176] Thanos Stathopoulos, John Heidemann, and Deborah Estrin. A remote code update mechanism for wireless sensor networks. Technical report, DTIC Document, 2003.

[177] William A Stein et al. Sage : open source mathematics software, 2008.

[178] Jean-Luc R Stevens, Marco Elver, and James A Bednar. An automated and reproducible workflow for running and analyzing neural simulations using lancet and ipython notebook. *Front Neuroinform*, 7 :44, 2013.

[179] El-Ghazali Talbi. *Metaheuristics : from design to implementation*, volume 74. John Wiley & Sons, 2009.

[180] Lu Tan and Neng Wang. Future internet : The internet of things. In *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, volume 5, pages V5–376. IEEE, 2010.

[181] 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.

[182] Serbulent Tozlu. Feasibility of wi-fi enabled sensors for internet of things. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, pages 291–296. IEEE, 2011.

[183] Sotirios A Tsaftaris. A scientist's guide to cloud computing. *Computing in Science & Engineering*, 16(1) :70–76, 2014.

[184] Hans van der Veer and Anthony Wiles. Achieving technical interoperability. *European Telecommunications Standards Institute*, 2008.

[185] Dimitri Van Heesch. Doxygen : Source code documentation generator tool, 2004.

[186] Lorenzo Vangelista, Andrea Zanella, and Michele Zorzi. Long-range iot technologies : The dawn of lora™. In *Future Access Enablers for Ubiquitous and Intelligent Infrastructures*, pages 51–58. Springer, 2015.

[187] Xavier Vilajosana, Pere Tuset, Thomas Watteyne, and Kris Pister. Openmote : Open-source prototyping platform for the industrial iot. In *Ad Hoc Networks*, pages 211–222. Springer, 2015.

[188] Klaus-Dieter Walter. Implementing m2m applications via gprs, edge and umts. *White paper—M2M Alliance*, 2009.

[189] 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.

[190] Yong Wang, Garhan Attebury, and Byrav Ramamurthy. A survey of security issues in wireless sensor networks. 2006.

[191] Jonathan Stuart Ward and Adam Barker. Observing the clouds : a survey and taxonomy of cloud monitoring. *Journal of Cloud Computing*, 3(1) :1–30, 2014.

[192] 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.

[193] 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.

[194] 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.

[195] Duane Wessels. *Web caching*. " O'Reilly Media, Inc.", 2001.

[196] 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.

[197] 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.

[198] 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.

[199] 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.

[200] Xiong Xiong, Kan Zheng, Rongtao Xu, Wei Xiang, and Periklis Chatzimisios. Low power wide area machine-to-machine networks : key techniques and prototype. *Communications Magazine, IEEE*, 53(9) :64–71, 2015.

[201] Jocelyn Young. Science interactive notebooks in the classroom. *Science Scope*, 26(4) :44–57, 2003.

[202] 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.

[203] Jerry Zhao, Ramesh Govindan, and Deborah Estrin. Residual energy scans for monitoring wireless sensor networks. In *IEEE WCNC*, 2002.

Passerelle intelligente pour réseaux de capteurs sans fil contraints

Rémy Léone

RESUME :

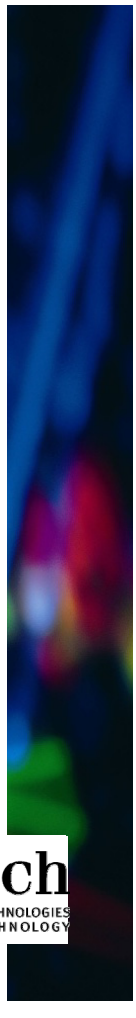
Les réseaux de capteurs sans fil (aussi appelés LLNs en anglais) sont des réseaux contraints composés de nœuds ayant de faibles ressources (mémoire, CPU, batterie). Ils sont de nature très hétérogène et utilisés dans des contextes variés comme la domotique ou les villes intelligentes. Pour se connecter nativement à l'Internet, un LLN utilise une passerelle, qui a une vue précise du trafic transitant entre Internet et le LLN du fait de sa position. Le but de cette thèse est d'exposer comment des fonctionnalités peuvent être ajoutées à une passerelle d'un LLN dans le but d'optimiser l'utilisation des ressources limitées des nœuds contraints et d'améliorer la connaissance de leur état de fonctionnement. La première contribution est un estimateur non intrusif utilisant le trafic passant par la passerelle pour inférer l'utilisation de la radio des nœuds contraints. La seconde contribution adapte la durée de vie d'informations mises en cache (afin d'utiliser les ressources en cache au lieu de solliciter le réseau) en fonction du compromis entre le coût et l'efficacité. Enfin, la troisième contribution est Makesense, un framework permettant de documenter, d'exécuter et d'analyser une expérience pour réseaux de capteurs sans fil de façon reproductible à partir d'une description unique.

MOTS-CLEFS : Internet des objets, Réseaux de capteurs, Cache, Supervision, Recherche reproductible.

ABSTRACT :

Low-Power and Lossy Network (LLN)s are constrained networks composed by nodes with little resources (memory, CPU, battery). Those networks are typically used to provide real-time measurement of their environment in various contexts such as home automation or smart cities. LLNs connect to other networks by using a gateway that can host various enhancing features due to its key location between constrained and unconstrained devices. This thesis shows three contributions aiming to improve the reliability and performance of a LLN by using its gateway. The first contribution introduce a non-intrusive estimator of a node radio usage by observing its network traffic passing through the gateway. The second contribution offers to determine the validity time of an information within a cache placed at the gateway to reduce the load on LLNs nodes by doing a trade-off between energy cost and efficiency. Finally, we present Makesense, an open source framework for reproducible experiments that can document, execute and analyze a complete LLN experiment on simulation or real nodes from a unique description.

KEY-WORDS : Internet of Things, Low-Power and Lossy Networks, Cache, Monitoring, Reproducible Research



TELECOM ParisTech

EDITE

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY