Routage efficace et garanti dans les réseaux de capteurs sans fil

Henry-Joseph Audéoud

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : **Informatique**

Arrêté ministériel : 25 mai 2016

Présentée par
Henry-Joseph AUDÉOUD,

dirigée par **Martin HEUSSE**,

préparée au sein du **Laboratoire d'Informatique de Grenoble**,

dans **l'École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique**.

Routage efficace et garanti dans les réseaux de capteurs sans fil

Thèse soutenue publiquement le **9 décembre 2019**, devant le jury composé de :

Anthony BUSSON
Professeur à l'Université Lyon 1, Rapporteur

Martin HEUSSE
Professeur à Grenoble INP, Directeur de thèse

Nathalie MITTON
Directrice de recherche à l'Inria Lille Nord Europe, Examinatrice

Thomas NOËL
Professeur à l'Université de Strasbourg, Rapporteur

Bernard TOURANCHEAU
Professeur à l'Université Grenoble Alpes, Président

Remerciements

Je tiens à remercier mon directeur de thèse, Martin HEUSSE, qui m'a accompagné dès la fin de l'ENSIMAG. Tu m'as fait confiance, et tu as su me laisser une grande liberté de travail et de pensée, me permettant d'avoir toutes sortes d'idées tout en m'aidant lorsqu'il le fallait à les formaliser et à les appliquer, guidant ainsi par tes conseils mes forces en direction du but à atteindre.

Je désire aussi remercier les membres de mon jury, Anthony BUSSON et Thomas NOËL, Nathalie MITTON et Bernard TOURANCHEAU. Vous avez étudié consciencieusement le travail que je présente ici, et vous avez même trouvé à m'encourager pendant la soutenance, alors que ma thèse touchait son terme.

Je pense beaucoup aussi à tous mes collègues. À toi, Pierre et tous ces bons moments passés ensemble dans notre bureau (D317 comme 415) ; à Timothy, mon cher « jumeau de thèse », et ces exercices de pensées qu'on mettait certainement trop souvent en pratique ; à Étienne, et tous ces moments partagés ensemble autour de WalT, du kernel, du *Linux Magazine* et d'un café. À tant d'autres aussi de l'équipe Drakkar : Mira, Baptiste, Élodie, Takwa, Andrzej, Franck, Olivier, Faten, Mickaël, Hai, Ulysse, et j'en passe, qui m'avez accompagné et continuez à le faire. Je pense à mes collègues d'enseignement également, et entre autres à Roland, Ghislaine et Christophe, qui avez été comme plusieurs autres mes enseignants à l'ENSIMAG avant d'y être mes collègues.

Je voudrais aussi remercier mes parents. Vous avez toujours pensé à moi dans vos prières et dans vos cœurs, et pas seulement dans les moments faciles. Ma petite sœur, aussi, tu as remplacé avantageusement ces trajets en train que je faisais au début de ma thèse, par des soirées qui, bien que souvent écourtées par mon retour tardif, étaient des moments d'un repos d'esprit oh combien profitable. Enfin Roselyne, ma femme, bien que tu sois arrivée chronologiquement parmi les dernières, tu n'as pas été la moindre à m'écouter, me consoler, m'encourager et prier.

Mais par-dessus tout, je pense à mon Dieu. Si des prières sont montées vers toi, tu as aussi su y répondre ; et tes réponses, pleines d'amour et de sagesse, ont été pour mon cœur un baume d'encouragement à nul autre comparable. Quel repos de savoir que, quelques soient les circonstances de mon chemin et l'état même de mon cœur, toi, à cause de toi-même, ô Dieu, TU DEMEURES FIDÈLE.

Résumé

Les réseaux de capteurs sans fil qui nous occupent dans cette thèse sont un ensemble d'appareils connectés les uns aux autres par des technologies bas débit et faible consommation. Leur rôle est de prendre des mesures sur l'environnement physique qui les entoure (suivis météoclimatiques, contrôles d'installations industrielles, relevés de l'état des réseaux de distributions, surveillance topographique…). Ces mesures doivent ensuite être collectées vers l'extérieur du réseau. Comme les capteurs ont une courte portée de communication radio, les transmissions sont faites en multisaut, les capteurs proches de la destination relayant l'information émise par ceux qui en sont plus éloignés. À cause du mouvement des nœuds eux-mêmes ou d'objets dans leur environnement perturbant les communications sans fil, la topologie exacte du réseau est sujette à des changements. De plus, les capteurs eux-mêmes, alimentés par batterie pour la plupart, sont limités en énergie et par là en capacité de transmission. Les techniques d'économie d'énergie appliquées pour éteindre la radio la plupart du temps imposent alors des contraintes de synchronisation supplémentaires.

Pour acheminer l'information dans le réseau, le protocole de routage établit des routes, de façon à ce que les capteurs puissent relayer l'information depuis et jusqu'au routeur de bordure du réseau à travers des liens fiables et conduisant jusqu'à la destination à travers des chemins courts. À cause des limitations des capteurs, le protocole de routage doit être efficace en énergie, c'est-à-dire que la surcharge des transmissions radio impliquées par le protocole de routage lui-même doit être aussi légère que possible. Il doit aussi être capable de rétablir la connectivité en cas de changement dans la topologie du réseau, et ce sans créer de boucles de routage pénalisant tant la qualité de service que les réserves d'énergie des nœuds.

Ce document décrit un protocole de routage répondant à ces objectifs. Il est capable de créer un arbre de collecte autoréparant permettant d'extraire l'information hors du réseau, ainsi que des routes pour distribuer des commandes ou des accusés de réception aux nœuds. Il valide aussi le chemin emprunté par chaque paquet transmis afin de garantir qu'ils n'entrent jamais dans une boucle de routage. Le protocole est mis en situation dans des simulations et aussi des expérimentations en plateforme réelle, montrant l'efficacité des mécanismes proposés.

Afin d'améliorer sa capacité à choisir les meilleurs liens disponibles, je propose également l'utilisation d'une nouvelle estimation de leur qualité. Elle est basée sur deux mesures complémentaires : une mesure à long terme du niveau de bruit ambiant présent sur le canal radio, et une mesure ponctuelle de la puissance du signal reçu de l'émetteur. Ces deux mesures fournissent une estimation du rapport signal à bruit, et par là du taux de réception attendu. Cette estimation est à la fois précise, rapide à obtenir, et adaptée aux contraintes des capteurs et des réseaux desquels nous parlons.

Mots-clefs Internet des objets, réseaux multisauts, IEEE 802.15.4, réseaux de capteurs sans fil, ad-hoc, protocoles de routage, LRP, RPL, AODV, LOADng, boucles de routage, efficacité énergétique, Contiki, FIT/IoT-LAB, expériences, simulation, estimation de la qualité des liens sans fil, proportion de paquets délivrés, rapport signal-à-bruit, mesures par échantillons.

English abstract

The wireless sensor networks that we work with in this thesis are a set of devices connected to each other by low-rate and low-power technologies. Their role is to produce measures on the physical environment around them (meteorological and climate condition tracking, monitoring of industrial installations, control of distribution grids, topographical surveillance...). These measures must then be collected out of the network. Since the sensors have short range radios, transmissions are multi-hop, the sensors close to the destination relaying the information transmitted by those which are further away from it. Because of the movement of the nodes themselves or of objects in their environment interfering with wireless communications, the exact topology of the network is subject to change. In addition, the battery-powered sensors are limited in energy and therefore in transmission abilities. The power-saving techniques applied to turn off the radio most of the time impose synchronization constraints.

To route information through the network, the routing protocol establishes routes, so that the sensors can relay information from and to the network border router through reliable links leading to the destination through short paths. Due to sensor limitations, the routing must be energy efficient, i.e. the overload of the radio transmissions involved by the routing algorithm itself must be as lightweight as possible. It must also be able to restore connectivity on a network topology change without creating routing loops that negatively impact the quality of service and the energy reserves of the nodes.

This document describes a routing protocol that meets these objectives. It is capable of creating a self-healing collection tree that extracts information out of the network, as well as from the routes to distribute command messages or acknowledgment to the nodes. It also validates the data path of each packet to ensure that they never enter a routing loop. The protocol is run in simulations and also on real platform experiments, showing the effectiveness of the proposed mechanisms.

In order to improve its ability to choose the best available links, I also propose the use of a new estimation of their quality. It is based on two complementary measurements: a long-term measurement of the ambient noise level on the radio channel, and a measurement of the power of the signal received from the transmitter. These two measurements provide an estimate of the signal-to-noise ratio, and thereby the expected reception rate. This estimate is both accurate, quick to obtain, and adapted to the constraints of sensors and networks we are talking about.

Keywords Internet of Things, multi-hop networks, IEEE 802.15.4, wireless sensor networks, adhoc, routing protocols, LRP, RPL, AODV, LOADng, routing loops, energy efficiency, Contiki, FIT/IoT-LAB, experiments, simulation, wireless link quality estimation, packet delivery ratio, signal to noise ratio, sampled measurement.

Table des matières

Liste des figures

Liste des tableaux

Glossaire & liste des acronymes

6LoWPAN

IPv6 over Low-Power Wireless Personal Area Networks, c.-à-d. IPv6 au-dessus de réseaux personnels sans fil de faible puissance. ix, 13–16, 60

6TiSCH

IPv6 over the TSCH mode of IEEE 802.15.4e, c.-à-d. IPv6 par-dessus le mode TSCH de IEEE 802.15.4e. ix, 14, 15, 20

6top

6TiSCH Operation Sublayer, c.-à-d. sous-couche d'opération de 6TiSCH. 15, *voir aussi* 6TiSCH

algorithme *Trickle*

« Algorithme permettant aux nœeuds d'échanger de l'information de façon robuste, efficace en énergie, simple, et passant à l'échelle sur un moyen de communication partagé avec pertes (p. ex. les LLN) » [30]. 20, 35, 50, 57, 62, 65

AODV

Ad-hoc On-demand Distance Vector, c.-à-d. protocole de routage ad hoc, à la demande, et de type vecteur de distances. 16, 18, 20–22, 30, 40, 42, 43, 46, 50–52, 54

arbre de collecte

Organisation sous forme d'arbre enraciné. ix, x, 19, 20, 30–36, 39, 40, 45–47, 50, 53, 57, *voir aussi* DODAG

AS

Autonomous System, c.-à-d. système autonome. 18, 21

ASK

Amplitude-Shift Keying, c.-à-d. modulation d'amplitude. 9–11

Babel

L'algorithme de routage Babel. 18, 21, 22, 41

bande ISM

Bande radio « industriel, scientifique et médical ». 7, 75

BATMAN

better approach to mobile ad hoc networking, c.-à-d. une meilleure approche pour les réseaux mobiles ad hoc . 21, 22

Bellman-Ford

Algorithme de calcul du plus court chemin entre deux nœuds sur un graphe. ix, 18, 32–34, 39, 50, 51

BER

Bit Error Rate, c.-à-d. taux d'erreur binaire. xi, 24, 88, *voir aussi* PER

BFD

Bidirectional Forwarding Detection, c.-à-d. détection de transmission bidirectionnelle. 53

BGP

Border Gateway Protocol, c.-à-d. Protocole des passerelles de bordures. 18, 21, 22

boucle de routage

État des tables de routage répartis sur les différents routeurs tel qu'un paquet retourne, au bout d'un certain nombre de transmissions, à un routeur qui l'a déjà routé. x, 2, 3, 17, 18, 21, 31–33, 35, 38, 40, 43, 46–49, 54, 59

BPSK

Binary Phase-Shift Keying, c.-à-d. modulation binaire de phase. 11

BRK

Break, c.-à-d. cassé. Message utilisé pour signaler une cassure (*a break*) dans l'arbre de collecte de LRP. x, 38–41, 43, 52, 53, 59, 61, 62, 67

CCA

Clear Channel Assessment, c.-à-d. évaluation d'un canal libre. 75, 76, 91, 93

CCDF

Complementary cumulative distribution function, c.-à-d. complémentaire de la fonction de répartition. En termes mathématiques, $CCDF_X(v) = 1 - CDF_X(v) = P(X \geq v)$. xii, 90, 92

CoAP

Constrained Application Protocol, c.-à-d. protocole d'application contraint. 15–17

Contiki

système d'exploitation pour réseau de capteur sans fil. 10, 16, 52, 56, 57, 59, 60

ContikiMAC

Couche MAC proposée par DUNKELS [90] et implémentée dans Contiki. 10, 12, 50, 60, 62, 63, 67, 87

couche application

7e couche du modèle OSI. 15

couche liaison de données

2e couche du modèle OSI, responsable de l'organisation et du contrôle du support de communication. 6, 9, 13, 15–17, 30, 42, 52, 62, 74, 76, 86, 90, 91

couche MAC

Sous-couche *Media Access Control*, c.-à-d. contrôle d'accès au support. 6, 10, 12, 13, 15, 16, 25, 30, 50, 52, 60, 62, 86

couche physique

1re couche du modèle OSI, responsable de la transmission effective des octets sur le canal de communication. xiii, 6, 9–13, 15, 16, 25, 31, 42, 60, 74, 75

couche réseau

3e couche du modèle OSI, responsable du routage et du relayage pour la construction de'une voie de communication de bout à bout. 6, 13, 15–17

couche transport

4e couche du modèle OSI. 15

CSL

Coordinated Sampled Listening, c.-à-d. écoute coordonnée de préambule. 10, 25, 63

CSMA/CA

Carrier Sense Multiple Access / Collision Avoidance, c.-à-d. écoute d'un support à accès multiple et à évidemment de collision. 91

CSS

Chirp Spread Spectrum, c.-à-d. étalement de spectre par chirps. 9–11

CTP

Collection Tree Protocol, c.-à-d. protocole à arbre de collecte. 42, 51

DAO

Destination Advertisement Object, c.-à-d. objet d'annonce de destination. 51, 57, 59, 62, 63

DAS

Débit d'Absorption Spécifique. 8

DIO

DODAG Information Object, c.-à-d. objet d'information DODAG. ix, 32–37, 39, 40, 42, 43, 47, 48, 50, 52, 54, 57, 59–63, 65, 67

DIO sollicité

DIO envoyé par un nœud S en réponse à un autre nœud N lorsque la route que N aurait en sélectionnant S comme successeur est meilleure que celle que N annonce avoir. 33, 61

DIS

DODAG Information Sollicitation, c.-à-d. sollicitation d'information DODAG. 33, 35, 60, 61, 65

DNS

Domain Name System, c.-à-d. système de noms de domaine. 13, 17

DODAG

Destination-Oriented Directed Acyclic Graph, c.-à-d. graphe sans cycles orienté vers une destination. 19, 33, 52, 65, 67, *voir aussi* arbre de collecte

Dotdot

Nouveau nom de la ZCL. 16, *voir aussi* ZCL

DQCSK

Differential Quadrature Chirp-Shift Keying, c.-à-d. modulation orthogonale différentielle de chirps. 11

DQPSK

Differential Quadrature Phase-Shift Keying, c.-à-d. modulation orthogonale différentielle de phase. 11

DSDV

Destination-Sequenced Distance Vector, c.-à-d. vecteur de distances séquencé par destination. 18, 21, 22, 32

DSR

Dynamic Source Routing, c.-à-d. routage dynamique à la source. 19, 21, 22, 40

DSSS

Direct Sequence Spread Spectrum, c.-à-d. étalement de spectre à séquence directe. 9

DTLS

Datagram Transport Layer Security, c.-à-d. sécurité de la couche transport en datagrammes. 16

DUAL

Diffusing Update Algorithm, c.-à-d. algorithme de mise à jour diffusantes. 21

EGP

Exterior Gateway Protocol, c.-à-d. protocole extérieur des passerelles. Protocole de routage utilisé entre les AS. 22

EIGRP

Cisco's Enhanced Interior Gateway Routing Protocol, c.-à-d. protocole de routage intérieur des passerelles amélioré. 21, 22, 24, 74

ERS

Expanding Ring Search, c.-à-d. recherche par anneau grandissant. Technique de recherche, où la portée de la recherche est limitée à une certaine taille, et étendue si aucun résultat n'est obtenu. 31, 40, 61

états de lien

Classe de protocole de routage où les nœuds connaissent toute la carte du réseau, et calculent chacun le chemin le plus court vers toutes les destinations. 18, 19, 21, 22

ETX

Expected Transmission count, c.-à-d. décompte des transmissions attendues. 23, 24, 62, 63, 74, 80, 94

EUI

Extended Unique Identifier, c.-à-d. identifiant unique étendu. 12

FCS

Frame Check Sequence, c.-à-d. séquence de contrôle de la trame. 9, 13, 88

FDMA

Frequency Division Multiple Access, c.-à-d. accès multiple par division en fréquence. 12

FHSS

Frequency Hopping Spread Spectrum, c.-à-d. étalement de spectre par sauts en fréquence. 12

G3-PLC

G3-Power Line Communication, c.-à-d. G3/communication par courants porteurs. 83

GFSK

Gaussian Frequency-Shift Keying, c.-à-d. modulation gaussienne de fréquence. 9–11

GTS

Guaranteed Time Slots, c.-à-d. créneaux temporels guarantis. 10, 12

HELLO

Message échangé entre les nœuds LRP pour estimer la qualité du lien dans les deux sens. 42, 51, 61–63, 65

HNA

Host and Network Associations, c.-à-d. associations réseau et hôte. 20

hop count

hop count, c.-à-d. décompte des sauts. 23, 33, 57, 61, 62, 83, 96

HRP UWB

High-Rate Pulse, c.-à-d. pulsations à rythme élevé. 11

HTTP

HyperText Transfert Protocol, c.-à-d. protocole de transfert hypertexte. 15

ICMP

Internet Control Message Protocol, c.-à-d. protocole de message de contrôle sur Internet. 15, 16

IEEE

Institute of Electrical and Electronics Engineers, c.-à-d. institut des ingénieurs électriciens et électroniciens. 6, 8, 16, 97

IESG

Internet Engineering Steering Group, c.-à-d. groupe directeur de l'ingénierie d'Internet. 18, *voir aussi* IETF

IETF

Internet Engineering Task Force, c.-à-d. groupe de travail de l'ingénierie d'Internet. 3, 13, 15, 16, 18, 35, 98

IGP

Interior Gateway Protocol, c.-à-d. protocole intérieur des passerelles. Protocole de routage utilisé à l'intérieur d'un AS. 22

IoT-LAB

Plateforme d'expérience ouvert à très grande échelle, déployée par l'établissement FIT (Future Internet Testing). xi, xiii, 31, 59–61, 65, 75, 76, 80, 84–87

IP

Internet Protocol, c.-à-d. protocole internet. 13, 17, 41, *voir aussi* IPv6

IPv6

Internet Protocol, version 6, c.-à-d. protocole internet en version 6. 13, 15–18, 41, 51–53, 60, *voir aussi* IP

IS-IS

Intermediate System to Intermediate System, c.-à-d. système intermédiaire à système intermédiaire, un protocole de routage IGP. 21, 22

LLN

Low-Power and Lossy Networks, c.-à-d. réseaux de faible puissance avec pertes. 12, 18, 22, 69

LOADng

Lightweight On-demand Ad-hoc Distance-vector Routing Protocol — next generation, c.-à-d. Protocole de routage léger, à la demande, ad hoc — nouvelle génération. 20–22, 30, 41, 42, 45, 50, 51, 54, 83

LPWAN

Low-Power Wide Area Network, c.-à-d. réseaux étendus de faible puissance . 96

LQI

Link Quality Indicator, c.-à-d. indicateur de la qualité du lien. xi, xiii, 24, 74, 75, 78, 80–85, 93

LRP

Lightweight Routing Protocol, c.-à-d. protocole de routage léger. x, 3, 30–33, 35, 36, 38, 40–43, 46–48, 50–54, 56–63, 65, 67, 69, 95

LRP UWB

Low Rate Pulse, c.-à-d. pulsations à rythme lent. 10, 11

MANET

Mobile Ad Hoc Networks, c.-à-d. réseaux mobiles ad hoc. 3, 19–22

mode RX

mode réception pour une radio. 10, 75, 87, *voir aussi* mode TX

mode TX

mode émission pour une radio. *voir aussi* mode RX

modèle OSI

Open Systems Interconnection, c.-à-d. systèmes d'interconnexion ouverts. ix, 6, 13, 17

monodiffusion

Envoi à un destinataire unique (*unicast*). 12, 33, 35, 39–42, 45, 50, 60, 61, 63, 65, 67, *voir aussi* multidiffusion

moyenne glissante exponentielle

Moyenne calculée sur des termes dont le poids de chacun décroît exponentiellement. 87

moyenne glissante pondérée

Moyenne calculée sur des termes dont le poids de chacun décroît linéairement. 87

MPR

Multi-Points Relays, c.-à-d. relais multipoints. 19, 21

MSK

Minimum Shift Keying, c.-à-d. modulation par décalage minimum. 11

MTU

Maximum Transmission Unit, c.-à-d. unité de transmission maximum. Taille maximale d'un PDU pouvant être transmis sans fragmentation sur une interface. 13, *voir aussi* PDU

multidiffusion

Diffusion à tous (*broadcast*). ix, 12, 13, 34–36, 39–43, 45, 50, 52, 61, 63, 65, 75, *voir aussi* monodiffusion

NHDP

Neighborhood Discovery Protocol, c.-à-d. protocole de découverte du voisinage. 42

NUD

neighbor unreachability detection, c.-à-d. détection de l'inaccessibilité des voisins. 30, 52, 53, 59

numéro de séquence

Numéro utilisé pour identifier un message. ix, 18, 22, 32, 34–41, 43, 45, 47–49, 52–54, 61

OLSR

Optimized Link State Routing Protocol, c.-à-d. Protocole de routage à états de lien optimisé. 19–22

OOK

On-Off Keying, c.-à-d. modulation allumé/éteint. 10

OpenWSN

Open Wireless Sensor Networks, c.-à-d. réseaux de capteurs sans fil à source ouverte, un projet de l'université de Californie, Berkeley. 15

O-QPSK

Offset Quadrature Phase-Shift Keying, c.-à-d. modulation orthogonale décalée de phase. xi, 9–11, 60, 75, 77, 89

OSPF

Open Shortest Path First, c.-à-d. chemin le plus court disponible d'abord, un protocole de routage pour IGP. 18, 21, 22

PDR

Packet Delivery Ratio, c.-à-d. proportion de paquets délivrés. 77, *voir aussi* PER

PDU

Protocol Data Unit, c.-à-d. unité de donnée de protocole. Correspond aux données spécifiques d'une couche (en-tête, en-queue,...) et aux informations des couches supérieurs (encapsulées). xi, 9, 13, 52, 76, 81, 90

PER

Packet Error Rate, c.-à-d. proportion de paquets perdus. xi–xiii, 24, 75–94, *voir aussi* PDR

PPM

Pulse Position Modulation, c.-à-d. modulation par pulses. 10

prédécesseur

Saut précédent sur la route par défaut le long de l'arbre de collecte, c.-à-d. celui qui a choisi un successeur. x, 35, 40, 44, 45, 48, 49, 53, *voir aussi* successeur

prochain saut

Nœud voisin à qui seront transmis les paquets suivant une route donnée. 17, 18, 32, 35, 43, 47–49

ratio d'activité cyclique

(*duty cycle*) rapport entre la durée d'activité de la radio dans un cycle et la durée de ce cycle. 7, 10, 16, 86, 87

RERR

Route Error, c.-à-d. erreur de route. 49, 50, 52, 59

réseau de capteurs sans fil

Wireless Sensors Network (WSN). 1–3, 8, 13, 15, 17–25, 32, 47, 52, 54, 63, 74

RFC

Requests For Comments, c.-à-d. demande de commentaires, documents publiés par l'IETF. 13, 18, 35, 42, 53, 54, 62, *voir aussi* IETF

RIP

Routing Information Protocol, c.-à-d. protocole d'information de routage. 16, 18, 21, 22

RIT

Receiver initiated Transmission, c.-à-d. transmission initiée par le récepteur. 12

RNSI

Received Noise Strength Indicator, c.-à-d. indicateur de la puissance du bruit reçu. xii, 75, 76, 87, 90, 92, *voir aussi* RSSI

ROLL

Routing Over Low power and Lossy networks, c.-à-d. Routage au-dessus des LLN, un groupe de travail de l'IETF. 3

RPL

Routing Protocol for LLN, c.-à-d. protocole de routage pour les LLN. x, 3, 15, 17–22, 30, 31, 33, 35, 43, 46, 50, 51, 53, 54, 56–63, 65, 67, 69

RREP

Route Response, c.-à-d. réponse de route. x, 43–49, 51–53, 57, 59, 61

RREQ

Route Request, c.-à-d. requête de route. x, 41–46, 51–54, 57, 59, 61, 62, 67

RSSI

Received Signal Strength Indicator, c.-à-d. indicateur de la puissance du signal reçu. xi, xii, 24, 74–76, 78–84, 87, 88, 92–94

SHR

Synchronization Header, c.-à-d. en-tête de synchronisation. 87

SNR

Signal-Noise Ratio, c.-à-d. rapport signal à bruit. xi, 9, 24, 77, 80, 85, 88–90

SPI

Serial Peripheral Interface, c.-à-d. interface périphérique série. 86

successeur

Prochain saut pour la route par défaut le long de l'arbre de collecte. ix, x, 32, 33, 35, 36, 38–41, 43–45, 47–50, 52, 53, *voir aussi* prédécesseur

TCP

Transport Control Protocol, c.-à-d. protocole de contrôle de transmissions. 13, 15, 16

TDMA

Time Division Multiple Access, c.-à-d. accès multiple par division en temps. 12

Thread

Pile protocolaire réseau développée par le *Thread Group*, et basé sur IEEE 802.15.4. ix, 14, 16

topologie

Ensemble de liens dans un réseau physiquement utilisable par les nœuds. 36

TORA

Temporally-Ordered Routing Algorithm, c.-à-d. Algorithme de routage temporellement ordonné. 19, 21, 22, 41

TPGF

Two-Phase geographic Greedy Forwarding, c.-à-d. Transmission géorgaphique gourmande à deux phases. 41

TSCH

Time-slotted Channel Hopping, c.-à-d. Saut de fréquence à créneaux temporels. 12, 15, 16, 25, 50, 87, 91

UDP

User Datagram Protocol, c.-à-d. protocole de datagramme utilisateur. 13, 15–17, 61, 62, 65

UIT

Union Internationale des Télécommunications. 7

UPD

Update, c.-à-d. mise-à-jour. Message utilisé pour réparer localement l'arbre de collecte de LRP. 38, 40, 41, 43, 45, 59, 61, 67

UWB

Ultra-Wide Band, c.-à-d. bande ultra-large. 8, 10, 11

vecteur de chemins

Classe de protocole de routage où les nœuds voisins échangent des informations détaillées décrivant toute la route pour le préfixe annoncé. 18, 22, *voir aussi* vecteur de distances

vecteur de distances

Classe de protocole de routage où les nœuds voisins échangent des informations aggrégée sur les routes. Aucun nœud ne possède une vision globale du réseau. 18, 19, 21, 22, 32, 51

WPAN

Wireless Personnal Area Networks, c.-à-d. réseaux personnels sans fil. 15

ZCL

Zigbee Cluster Library, c.-à-d. bibliothèque de groupe Zigbee, une bibliothèque logicielle Zigbee. 16, *voir aussi* Dotdot

Zigbee

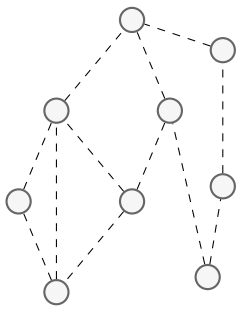
Ensemble de protocole de la communauté industrielle Zigbee Alliance, s'appuyant sur la norme IEEE 802.15.4. ix, 14–16

ZRP

Zone Routing Protocol, c.-à-d. Protocol de routage par zones. 19, 21, 22

Introduction

Dans un réseau de communication, des nœuds communiquent en s'échangeant des paquets d'information. Lorsque deux nœuds ne sont pas directement connectés, ils doivent s'aider des nœuds intermédiaires pour acheminer les paquets qu'ils désirent s'échanger, formant ainsi des transmissions dites multisauts. Ce travail d'acheminement est une des deux parties du routage ; mais pour arriver à faire ce travail, il faut au préalable que les liens empruntés pour atteindre la destination aient été choisis. Il faut donc découvrir, établir et maintenir des routes le long desquelles les paquets de données pourront être acheminées. La nécessité d'avoir des algorithmes de routage automatiques dédiés à cette tâche vient du fait que les réseaux sont souvent bien trop grands pour être appréhendés dans leur globalité par l'esprit humain, que les calculs pour trouver la meilleure route parmi celles disponibles est une tâche fastidieuse, et que le temps de réaction à une modification du réseau doit être réduit au minimum pour augmenter la qualité de service.



Cette thèse se place dans le contexte du routage dans les réseaux de capteurs sans fil (abbrégé WSN, *Wireless Sensor Networks*). Dans ces réseaux, les nœuds appelés capteurs sont des systèmes embarqués capables de mesurer ou relever quelque information du monde physique qui les entoure. Ces données générées doivent ensuite être transmises à travers leurs modules de transmission sans fil jusqu'à un point de collecte du réseau où un nœud spécial, appelé routeur de bordure, ou encore puits du réseau (*sink*), extrait ces données en les communiquant sur d'autres réseaux de plus grande envergure auxquels il est connecté, jusqu'à l'endroit où elles seront stockées, travaillées, utilisées. La faible portée des modules de transmission radio fait que les réseaux sont multisauts, et qu'il est nécessaire d'y offrir un service de routage. En plus des problématiques générales du routage dans un réseau, le routage en réseaux de capteurs sans fil met l'accent sur deux points : il s'agit de réseaux **sans fil**, et de réseaux **de capteurs**. Parmi les réseaux sans fil, nous pouvons citer la téléphonie sans fil, ou le WiFi, des réseaux que nous utilisons tous les jours. Parmi les réseaux de capteurs,

nous pouvons citer les réseaux de compteurs Linky, relevant des informations d'état du réseau à l'intérieur de la grille de distribution d'électricité. Les réseaux de capteurs sans fil combinent ces deux aspects.

Sur les réseaux filaires classiques, un commutateur fait de la retransmission des trames sur ses liens. De proche en proche, les paquets sont ainsi acheminés jusqu'à la destination dont l'adresse est lue dans la trame. Les routeurs utilisent des algorithmes distribués pour calculer les plus courts chemins sur des graphes bien plus grands. Les réseaux sans fil ajoutent cette première complexité en ce qu'ils utilisent un canal de diffusion, c'est-à-dire que, pour communiquer, un nœud émet un signal perçu par tous ses voisins, même si la trame n'est destinée qu'à l'un d'entre eux aux alentours. C'est ce qui fait déjà la différence entre un concentrateur (*hub*) et un commutateur (*switch*) dans les réseaux filaires. Ainsi, le domaine de collision entre les trames est aussi large que la portée de la radio du nœud émetteur, et les nœuds se perturbent donc l'un l'autre en transmettant — et non seulement pour les nœuds d'un même réseau, mais également entre les différentes technologies lorsque celles-ci coexistent dans la même zone géographique et la même bande de fréquences. Une deuxième complexité apportée par les liens sans fil est que, étant moins clairement définis, ils sont beaucoup moins stables que des liens filaires. Le mouvement d'un nœud modifiera les caractéristiques des liens le reliant à ses voisins ; et, même si les nœuds eux-mêmes ne bougent pas, de simples mouvements dans l'environnement sont suffisants pour modifier la propagation des ondes utilisées pour la transmission et obtenir le même effet néfaste. Ainsi, les réseaux sans fil sont des réseaux avec une grande variabilité topologique.

La troisième complexité apportée par ces plateformes sans fil est que les capteurs doivent être alimentés en énergie par batterie. Les réserves d'énergie doivent être suffisantes pour faire fonctionner le système de mesure lui-même, toute la partie de calcul (processeur, mémoire…) et le module de transmission. L'énergie est certes parfois extraite de l'environnement (p. ex. avec un capteur photovoltaïque), mais elle n'est pas forcément extraite en quantité très importante, et le capteur est alors à puissance limitée plutôt qu'à énergie limitée. De plus, on cherche à réduire au maximum le coût financier de production des capteurs. Ces contraintes, énergétiques et économiques, apportent des limitations, et font de ces plateformes des systèmes très restreints en ressources. Leur matériel électronique n'est pas forcément des meilleurs, ils ont en général des caractéristiques processeur et mémoire restreintes, et le module de transmission radio, le plus gros consommateur énergétique, doit souvent être éteint.

Toute cette complexité demande des protocoles de routage adaptés. Ce qu'il nous faut est un protocole de routage

— capable d'**extraire efficacement les données** produites par les capteurs en faisant des choix judicieux des liens choisis pour établir les routes. La possibilité de distribuer des commandes à l'intérieur du réseau devra également être maintenue ;

— capable de fonctionner en émettant une quantité très restreinte de trafic, étant ainsi **efficace en nombre de messages de contrôle émis**. Cela est d'autant plus valable pour les messages multidiffusés, certes absolument nécessaires au routage, mais également plus coûteux lorsque les techniques d'économie d'énergie pour allonger la durée de vie des capteurs sont déployés au niveau radio ;

— adapté à la variablilité des liaisons sans fil, et donc capable de s'autoréparer rapidement à l'échelle de la génération des données, pour **garantir la connectivité montante et descendante** avec le routeur de bordure, et ne laissant jamais aucun nœud complètement déconnecté du réseau ;

— offrant également des **garanties au sujet de l'acheminement des données** dans le réseau, pour s'assurer qu'ils ne sont jamais conduits dans une boucle de routage qui impacte tant

la qualité de service que les réserves énergétiques des nœuds.

Beaucoup de protocoles de routage pour réseaux mobiles ad hoc, ou MANET (*Mobile Ad Hoc Networks*), ont été proposés. Mais, quoiqu'ils prennent correctement en compte les contraintes de mobilité des réseaux sans fil, ils ne sont pas adaptés aux contraintes particulières des réseaux de capteurs. Le groupe de travail ROLL (*Routing Over Low power and Lossy networks*) [121] de l'IETF (*Internet Engineering Task Force*) a commencé à travailler sur cette problématique en 2008, pour proposer le protocole de routage RPL (*Routing Protocol for LLN*) [32] en 2012. Bien que conçu spécifiquement pour des réseaux à faible puissance, j'ai observé que RPL se révèle produire toujours une quantité de trafic de contrôle importante. LRP (*Lightweight Routing Protocol*), le protocole de routage étudié et présenté dans cette thèse, présente une approche différente, qui s'efforce de répondre au mieux aux objectifs ci-dessus.

Après avoir présenté plus en détails le contexte dans lequel le travail de cette thèse a été réalisé et les technologies associées déjà existantes (chapitre 1), nous aborderons le protocole de routage LRP, que j'ai étudié et sur lequel j'ai travaillé. C'est un protocole de routage pour réseaux de capteurs sans fil qui vise une surcharge légère, même lorsque des changements de topologies nécessitent une mise à jour des informations de routage. Il est de plus démontré comme étant sans boucle de routage à quelqu'instant que ce soit, à l'aide d'une vérification de l'acheminement à chaque fois qu'un paquet doit être commuté. Ce protocole est présenté dans ces mécanismes théoriques (chapitre 2) tout aussi bien qu'en observant son comportement pratique à travers des simulations et des expérimentations en plateforme réelle (chapitre 3).

La faible surcharge du protocole de routage lui impose, plus encore que d'autres comme RPL, d'être capable d'estimer précisément la qualité d'un lien radio alors que peu d'échanges ont été effectués dessus, idéalement après la réception d'un unique paquet — le cas d'un lien avec un voisin qui vient d'être découvert, mais aussi le cas de l'entretien de l'information sur la qualité du lien sans y nécessiter la transmission de beaucoup de trafic. J'ai étudié ce problème, voisin à celui du routage efficace, et proposé une méthode y répondant qui est abordé dans le dernier chapitre (chapitre 4).

Chapitre 1

État de l'art

Le modèle OSI (*Open Systems Interconnection*) découpe les tâches nécessaires pour la transmission d'information en plusieurs couches, avec plusieurs niveaux d'abstraction différents (cf. figure 1.1). La couche physique, 1re couche du modèle OSI, permet la transmission effective d'une chaîne de bits d'une machine à une autre à travers le support physique les reliant (câble électrique, fibre optique, ondes radio…). La couche liaison de données, ou couche MAC (*Media Access Control*), 2e couche, a ensuite un rôle d'organisation et de contrôle de la transmission sur ce canal de communication. Elle s'occupe du cadrage des trames, c'est-à-dire de passer d'une suite de bits à une suite de trames ; elle régule les émissions sur le support en cas d'accès multiple ; elle gère l'adressage physique des trames (désignation d'un hôte dans l'ensemble des machines accessibles) et, selon les cas, leur commutation à l'intérieur du réseau local ; elle contrôle les files d'attente de transmission ; elle fournit un contrôle et parfois une correction des erreurs, un acquittement des trames échangées… La couche réseau, 3e couche, permet ensuite la communication de bout en bout à travers différents réseaux locaux, en offrant un adressage global ; elle a ainsi le rôle du routage (calcul du meilleur chemin reliant les deux machines) et du relayage (transmission effective d'un paquet de données de sa source jusqu'à sa destination). Les couches 4, 5 et 6 organisent les transmissions en flux multiplexés, récupèrent les pertes de paquets, synchronisent les transmissions, chiffrent et structurent les données, en fonction des besoins définis par l'application, éponyme de la 7e couche.

Le travail de ma thèse porte principalement sur les trois couches basses de ce modèle. Après avoir abordé quelques aspects de la transmission physique des signaux en section 1.1, nous nous pencherons sur la norme IEEE 802.15.4 de l'IEEE (*Institute of Electrical and Electronics Engineers*) pour les réseaux sans fil à bas débit, en section 1.2. Cette norme décrit toute l'architecture de transmission, c'est-à-dire les couches physique et MAC ; des technologies utilisent ensuite ces deux couches pour implémenter une pile protocolaire complète, que nous aborderons également. Ensuite, dans la section 1.3, nous verrons plus en détails les tenants et les aboutissants inhérents au travail de routage. Enfin, en section 1.4.1, nous étudierons quelques métriques permettant d'estimer la qualité d'un lien, information nécessaire pour offrir un routage efficace.

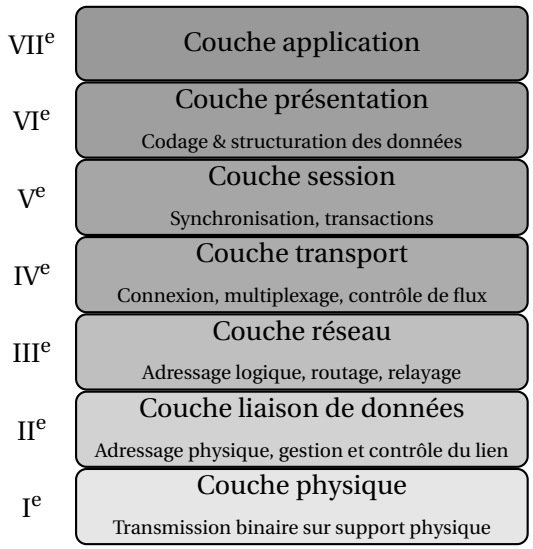


FIGURE 1.1 – Les 7 couches du modèle OSI.

1.1 Approche brève de la transmission du signal

Lorsque deux machines communiquent, un signal est envoyé à travers le support de communication. Ce signal transporte l'information que ces machines veulent se transmettre, c'est-à-dire finalement une suite de valeurs de bits. Les caractéristiques du signal peuvent être très diverses, en fonction des contraintes fixées par le contexte et les objectifs choisis.

Capacité maximale théorique d'un canal Le théorème de Shannon–Hartley, donne la capacité C d'un canal de communication, c'est-à-dire le débit d'information D_b maximal que ce canal pourra transmettre, en fonction de la bande passante B occupée et en présence de bruit blanc additif gaussien de puissance N, et avec un signal de puissance moyenne S :

$$C = B \times \log_2\left(1 + \frac{S}{N}\right) \tag{1.1}$$

D'après Hartley [51], le facteur $\sqrt{1 + S/N}$ est le nombre d'états distinguables d'un signal [1]. Lorsque la puissance du signal le permet, il est possible d'utiliser des signaux physiques composés de plus de deux états — un signal à M états transmettra $\log_2(M)$ bits par symbole. La bande passante occupée dépend directement du débit symbole, exprimé en bauds (Bd) : $B \geq 1/2 \times D_s = {}^1/_2 \times D_b \times \log_2(M)$.

Suivant les paramètres utilisés pour la communication, le débit binaire effectif D_b sera plus ou moins proche de cette limite. Le deuxième théorème de Shannon indique que, lorsque $D_b < C$, il existe un codage permettant de transmettre l'information avec une probabilité d'erreur arbitrairement petite. Inversement, les données ne seront pas transmises de façon fiable au-delà de la capacité du canal (lorsque $D_b > C$).

En pratique, à cause de leur variabilité temporelle, on ne peut compter que sur une estimation grossière des variables de l'équation 1.1. Le choix de D_b devra donc être fait suffisamment proche de C pour avoir une transmission efficace, et en même temps pas trop proche de C afin de garder un taux d'erreur bas.

Régulations Dans le cadre de transmissions sans fil, des régulations existent pour éviter un usage abusif de la ressource fréquentielle partagée. L'UIT (Union Internationale des Télécommunications) et des services affectataires locaux s'occupent de l'assignation des bandes de fréquences en fonction des besoins. Les autorités restreignent ainsi les bandes utilisables à un type d'utilisation, limitent la puissance d'émission des appareils, leur ratio d'activité cyclique (*duty cycle*), ou imposent l'utilisation de mécanismes d'écoute avant émission.

La bande ISM (industriel, scientifique et médical) est une bande de fréquences entre 2400 et 2500 MHz [2]. Elle est prévue pour les applications industrielles, scientifiques, médicales, domestiques ou similaires. Elle est aujourd'hui beaucoup utilisée dans ce cadre pour chauffer la matière, comme pour la thermothérapie, ou dans les fours à micro-ondes. Les applications de radiocommunication et de radiorepérage, bien qu'initialement exclues, y sont tolérées sous l'application de la directive RED [138]. Ils forment de nos jours la majeure partie de l'utilisation de cette bande. Le WiFi (exposé dans la norme IEEE 802.11 [11] — j'y référerai sous ce nom dans ce manuscrit), le

1. Ici avec S et N exprimées en puissance. Hartley utilisa ces grandeurs exprimées en termes de tensions, A étant l'amplitude maximale du signal et ΔV la précision de la mesure, sous la forme $1 + {}^A/_{\Delta V}$.

2. La bande des 2450 MHz n'est pas la seule bande de fréquence appelée ISM. Il existe d'autres bandes, comme la bande des 433 MHz, la bande des 915 MHz en Amérique ou des 868 MHz en Europe. La bande des 2450 MHz est parmi les plus larges (avec 100 MHz de largeur) et disponible partout dans le monde, et on parle souvent d'elle par abus de langage en parlant de bande ISM.

Bluetooth (temporairement exposé par la norme IEEE 802.15.1, puis développé par le groupe d'intérêt spécial Bluetooth), et encore bien d'autres appareils (téléphones sans fil, télécommandes de garage ou de voiture, drones,...) l'utilisent, ainsi aussi que IEEE 802.15.4 sur lequel nous revenons dans la section suivante.

Il s'agit donc d'une bande partagée, où des interférences entre les transmissions de ces technologies sont attendues. Cette thèse n'a pas comme objectif d'étudier les interactions entre elles. L'IEEE a par exemple déjà étudié la coexistence entre IEEE 802.15.4 et les autres technologies qu'il normalise, et entre autres sur cette bande des 2450 MHz [9]. Le groupe de travail IEEE 802.15.2 [7] étudie également la coexistence avec d'autres appareils sans fil dans les bandes non licenciées. Une étude succincte est menée au chapitre 4 pour quantifier dans une certaine mesure l'interférence à l'échelle d'une trame, particulièrement entre le WiFi (IEEE 802.11) et IEEE 802.15.4.

Puissance du signal Il est possible d'augmenter la puissance de transmission du signal, c'est-à-dire d'agir sur la variable S du théorème de Shannon–Hartley (équation 1.1), qui prédit alors une augmentation du rapport C/B — augmentation de la quantité d'information transmissible pour une bande fixée, ou la diminution de la bande consommée pour une quantité d'information transmissible fixée. Cela a cependant deux effets négatifs.

Le premier est l'effet de ces transmissions sur l'environnement. Au niveau des interférences avec d'autres communications, là où la puissance S du signal augmente pour une transmission, le niveau de bruit N augmente, peut-être légèrement, mais quand même pour toutes les transmissions partageant le même canal. Pire, les échos de l'onde transmise par l'appareil lui-même perturbent et déforment son propre signal, ou produisent de très forts évanouissements de canal parce que les ondes qui ont emprunté un autre chemin arrivent déphasées par rapport au signal. C'est ce que cherche à représenter le modèle de canal de Rayleigh. Les ondes transmises peuvent également avoir des impacts sur d'autres systèmes que des systèmes communicants, comme des radiographies médicales ou les systèmes de radioguidages à bord des avions, d'où les panneaux interdisant l'utilisation des téléphones portables à ces endroits. Des régulations locales peuvent aussi interdire de transmettre au-delà d'une certaine puissance. Un exemple de la vie courante est le DAS (Débit d'Absorption Spécifique) limité à 2 W/kg pour la vente d'appareils radioélectriques dans l'union européenne, ici à cause des effets de ces émissions sur le corps humain de l'utilisateur de l'appareil.

En plus de ces contraintes extérieures à l'appareil lui-même, l'augmentation de la puissance d'émission implique directement une augmentation de la puissance consommée par le module radio. Quand bien même cela n'est pas toujours équivalent à une consommation d'énergie proportionnelle — le débit étant plus élevé, l'émission est plus brève — c'est une option qui n'est pas toujours préférée dans les réseaux de capteurs sans fil (abbrégé WSN, *Wireless Sensor Networks*) à énergie limitée.

Bande passante Une autre possibilité est d'augmenter la bande passante (c'est-à-dire d'agir sur la variable B de l'équation 1.1). Là encore, l'avantage n'est pas forcément évident. Lorsque le bruit est blanc (c.-à-d. indépendant de la fréquence), doubler la bande passante revient également à doubler le niveau de puissance du bruit (variable N), et dans le cas de faibles puissances (p. ex. lorsque de l'étalement de spectre est utilisé), le gain est nul et la capacité du canal reste constante.

L'utilisation de bandes de fréquences plus grandes — jusqu'aux bandes UWB (*Ultra-Wide Band*), supérieures à 500 MHz — peut aussi servir à effectuer des mesures de distances (*ranging*) dans le réseau entre les appareils, et permettre ainsi de localiser un émetteur.

Il est aussi possible d'utiliser un signal dont la bande passante est plus grande que le débit binaire. Cette technique s'appelle **l'étalement de spectre**. Son intérêt est qu'elle permet d'utiliser

un rapport signal à bruit (ratio S/N) inférieur à 1, c'est-à-dire d'émettre à une puissance inférieure à celle du bruit en gardant un signal décodable. On peut ainsi fiabiliser une communication à très basse puissance.

Une des possibilités pour cela est de transmettre un seul bit d'information non pas à l'aide d'un seul symbole, mais à l'aide de plusieurs symboles consécutifs (appelés des *chips*), comme avec DSSS (*Direct Sequence Spread Spectrum*) où chaque bit est représenté par une séquence pseudoaléatoire prédéfinie de chips. C'est l'inverse d'une modulation utilisant des symboles avec un alphabet de taille supérieure à deux, où la bande passante consommée est alors réduite. L'information transmise est ici fiabilisée parce que l'énergie utilisée pour transmettre un bit est ainsi augmentée d'un facteur correspondant au nombre de chips utilisés pour coder ce bit. D'autres techniques d'étalement de spectre, comme avec CSS (*Chirp Spread Spectrum*), utiliseront des *chirps*, c'est-à-dire des signaux pseudopériodiques modulés linéairement autour d'une porteuse.

Modulation Il y a plusieurs façons de moduler, c'est-à-dire de modifier les paramètres du signal en fonction des données à envoyer. (en modulant p. ex. l'amplitude, la phase, ou la fréquence du signal). Le choix entre chaque technique est encore une question de compromis. Les modulateurs et démodulateurs ASK (*Amplitude-Shift Keying*) sont par exemple relativement bon marché et facile à faire — un point particulièrement utile dans les réseaux de capteurs où certains capteurs doivent être peu coûteux. La modulation O-QPSK (*Offset Quadrature Phase-Shift Keying*) est une modulation plus complexe que la modulation QPSK, mais elle a l'avantage d'offrir une enveloppe constante (pas de retour à zéro), ce qui permet l'utilisation d'amplificateurs électroniques à saturation, bien plus faciles à construire que des amplificateurs linéaires. La modulation GFSK (*Gaussian Frequency-Shift Keying*) utilise une modulation en fréquence filtrée ensuite par un filtre Gaussien pour réduire la puissance dans les bandes latérales et augmenter ainsi l'efficacité spectrale (c.-à-d. la quantité d'information envoyée dans une bande donnée).

1.2 La norme IEEE 802.15.4

La norme IEEE 802.15.4, initialement écrite en 2003, puis révisée en 2006, 2011 et 2015, se décrit elle-même comme définissant « le protocole et l'interconnexion compatibles pour les appareils de communication de données utilisant des transmissions radio fréquences à faible débit de données, de faible puissance et de faible complexité à courte portée dans un réseau personnel sans fil. » [10] Elle définit les caractéristiques des couches physique telles que présentées dans la section précédente, ainsi que la couche liaison de données qui gère ces couches physique. Après leur description dans les sections 1.2.1 et 1.2.2, la section 1.2.3 présente quelques utilisations de IEEE 802.15.4 en présentant des implémentations et des couches développées spécialement pour réseaux de capteurs au-dessus de celles définies dans la norme.

1.2.1 Couches physiques

L'objectif final de la couche physique est de contrôler l'émetteur-récepteur radio afin de communiquer, via des ondes radio modulées, avec les autres émetteurs-récepteurs pour transmettre la suite de bits, appelée PDU une fois cadrée par la couche liaison de données sus-jacente. IEEE 802.15.4 ne définit pas qu'une seule couche physique, mais « diverses couches physique [… pour couvrir] une grande variété de bandes de fréquences » [10] ; l'interface entre ces deux couches est cependant la même quelque soit la couche physique utilisée, et définit entre autres la taille d'un PDU (*Protocol Data Unit*) à 127 octets ou encore la taille du FCS (*Frame Check Sequence*) à 2 octets. Le tableau 1.1 regroupe ces couches physique, avec quelques unes de leurs

caractéristiques.

Cette diversité a l'avantage de mettre plusieurs supports physiques de communication, avec leurs points forts respectifs, à disposition de l'implémentation. Comme nous l'avons déjà vu, les couches physique ASK ou O-QPSK[3] peuvent avoir des avantages au niveau de leur coût de mise en place. À l'inverse, les couches physiques UWB nécessiteront peut-être un matériel un peu plus coûteux, mais offriront de faire de l'évaluation de distances (*ranging*). La couche physique LRP UWB (*Low Rate Pulse*), une couche UWB, propose deux modulations ayant chacune son propre avantage : soit une modulation OOK (*On-Off Keying*) pour proposer un débit élevé (jusqu'à 1 Mbit/s pour le mode dit *de base*, quatre fois plus élevé que la couche physique O-QPSK dans la bande des 2450 MHz), soit une modulation PPM (*Pulse Position Modulation*) dans le mode dit *à longue portée* pour permettre des communications avec de meilleures sensibilités. La couche physique GFSK, qui n'utilise pas d'étalement de spectre, présente une bande de fréquences très étroite en comparaison des autres (100 kHz, le filtrage Gaussien qui suit la mise en forme ensuite la réduit encore à 50 kHz) — et bande étroite veut aussi dire une puissance de bruit capturée plus petite — et atteint ainsi une efficacité spectrale élevée. CSS a l'avantage d'être résistant au multitrajet des ondes radio, et aussi à l'effet Doppler.

1.2.2 Couche MAC

La couche MAC a un rôle d'organisation des transmissions au-dessus de la couche physique sous-jacente (cf. figure 1.2). Dans le cas de la norme IEEE 802.15.4, celle-ci nous donne clairement sa portée :

> Les fonctions de la sous-couche MAC sont la gestion de balises (*beacon*), l'accès aux canaux, la gestion de créneaux temporels garantis (*Guaranteed Time Slots*), la validation des trames, la livraison de trames acquittées, l'association et la dissociation. De plus, la sous-couche MAC fournit des interfaces (*hooks*) pour la mise en œuvre de mécanismes de sécurité adaptés à l'application. — IEEE 802.15.4-2015 [10]

Modes de transmission La norme propose plusieurs modes de communication, dont l'optique finale est d'éteindre dans certains cas la radio des nœuds — ce sont en cela des techniques d'économie d'énergie. Le plus simple est le mode **sans balise** (*nonbeacon-enabled*), où les hôtes communiquent les uns avec les autres sans synchronisation temporelle. Cela permet une grande économie d'énergie pour les nœuds terminaux (qui ne routent pas le trafic d'autres nœuds) ; mais en contrepartie, cela nécessite que les nœuds dans le réseau soient constamment disponibles en mode RX (mode réception pour une radio) — il n'y a pour eux aucune économie d'énergie.

Le mode CSL (*Coordinated Sampled Listening*) — à **échantillonnage de préambules** — est un mode lui aussi pratiquement asynchrone. Ce mode permet au récepteur d'éteindre sa radio la plupart du temps (utilisation d'un ratio d'activité cyclique, où il se réveille périodiquement pour sonder le canal physique). Cet avantage est au prix d'un coût de transmission plus élevé du point de vue de l'émetteur. Lorsqu'un capteur transmet, il émet sa trame de façon répétitive (ou bien un préambule à la trame, indiquant un instant de rendez-vous pour une trame à venir), suffisamment longtemps pour que le récepteur se réveille durant ces transmissions et réponde par un acquittement. Un exemple est celui de PS-ALOHA [63]. Pour diminuer le nombre de répétitions du paquet, l'émetteur peut aussi se synchroniser sur le cycle de réception du nœud voisin, afin d'émettre le paquet suivant plus proche de l'instant de réveil de celui-ci. C'est ce que fait entre autres ContikiMAC [90], implémenté dans le système d'exploitation pour réseau de capteurs Contiki.

3. Dans IEEE 802.15.4, la modulation utilisée a souvent donné son nom à la couche physique elle-même.

TABLE 1.1 – Quelques couches physique telles que définies dans IEEE 802.15.4.

Couche physique	Version de la norme	Bande de fréquence	Nombre de canaux	Espace intercanal	Bande passante	Débit		Puissance d'émission	Sensibilité
		MHz		kHz	kHz	kchip/s	kbit/s	dBm	dBm
BPSK	2003	868	1	—	600	300	20	≥–3	≤–92
		915	10	2000	1200	600	40		
	2011[a]	950	8/2	600/200	600	300	20		
O-QPSK	**2003**	**2450**	**16**	**5000**	**3000**	**2000**	**250**	≥–3	≤–85
	2006	868	1	—	600	400	100		
	2006	915	10	2000	1500	1000	250		
	2011	780	8		1800				
	2015	2380	15	5000	1500				
ASK	2006[a]	868	1	—	480	400	250	≥–3	≤–85
		915	10	2000	1920	1600			
GFSK	2011[a]	950	12	400	50	—	100	≥–3	≤–85
	2015	920	n. d.	n. d.					
MSK	2015	443	15	108	47	—	31,25	≥–3	n. d.
				324	150	—	100		
				540	375	—	250		
		2450	42	2000	375	—	250	≥–13	n. d.
CSS	2011	2450	14	5000	15×10^3	1333[b]	250/1000	≥–3	≤–91
HRP UWB	2011	250–750	1	—	500×10^3	500×10^3	110 à $27{,}2 \times 10^3$	pas de minimum	n. d.
		3244–4742	3/1	500×10^3	500×10^3				
		5944–10234	8/1/2	$(500/1500) \times 10^3$	$(500/1000/1300) \times 10^3$				
LRP UWB	2015	6289–9185	3		≫1000	1000	1000	n. d.	n. d.
							250		
					≫2000	2000	31,25		

[a] Marqué comme déprécié, ou supprimé dans la version 2015 de la norme.
[b] Il s'agit des chips de la modulation DQPSK, qui seront ensuite suivis par un étalement CSS pour produire des symboles DQCSK, à un rythme de 166,6 kchirp/s.
n. d. : Non défini par la norme.

Glossaire :

ASK *Amplitude-Shift Keying*, c.-à-d. modulation d'amplitude.

BPSK *Binary Phase-Shift Keying*, c.-à-d. modulation binaire de phase.

CSS *Chirp Spread Spectrum*, c.-à-d. étalement de spectre par chirps.

DQCSK *Differential Quadrature Chirp-Shift Keying*, c.-à-d. modulation orthogonale différentielle de chirps.

DQPSK *Differential Quadrature Phase-Shift Keying*, c.-à-d. modulation orthogonale différentielle de phase.

GFSK *Gaussian Frequency-Shift Keying*, c.-à-d. modulation gaussienne de fréquence.

HRP *High-Rate Pulse*, c.-à-d. pulsations à rythme élevé.

LRP *Low Rate Pulse*, c.-à-d. pulsations à rythme lent.

MSK *Minimum Shift Keying*, c.-à-d. modulation par décalage minimum.

O-QPSK *Offset Quadrature Phase-Shift Keying*, c.-à-d. modulation orthogonale décalée de phase.

UWB *Ultra-Wide Band*, c.-à-d. bande ultra-large.

Un autre mode de la norme IEEE 802.15.4 propose l'utilisation de **balises** (*beacon-enabled*), permettant de synchroniser temporellement les appareils. L'émission de la balise définit trois périodes temporelles. La première, appelée période avec contention d'accès, permet à n'importe quel nœud de transmettre à l'émetteur de la balise (le coordinateur), mais avec gestion de la contention (entre autres, les nœuds doivent écouter avant de transmettre pour éviter d'écraser une transmission en cours). Les nœuds utilisent également cette période pour recevoir les messages que le coordinateur a indiqué avoir, en sollicitant explicitement l'envoi des trames grâce au mode de transmission RIT (*Receiver initiated Transmission*). Ensuite, la période dite sans contention, où sont autorisés à parler exclusivement les nœuds décrits dans la balise comme tels — ces nœud possèdent alors un GTS (*Guaranteed Time Slots*) créneau garanti pour envoyer une information. Enfin, la dernière période est une période où les radios sont éteintes afin d'économiser l'énergie — il n'y a pas de transmission. La longueur de ces périodes est configurable par l'émetteur de la balise.

Le mode **TSCH (*Time-slotted Channel Hopping*)** est également un mode synchronisé par l'envoi régulier de balises. Il hérite directement de la norme WirelessHART, développée séparément de IEEE 802.15.4. Il est intégré à celle-ci par l'amendement IEEE 802.15.4e. Il a comme objectif de proposer une interface déterministe et robuste pour des applications industrielles. Il synchronise les nœuds par l'allocation de cellules définies en temps et en fréquence. Les transmissions sont ordonnancées par des allocations précises de créneaux temporels, mais aussi fréquentiels, combinant ainsi un accès multiple divisé en temps (TDMA) et en fréquence (FDMA (*Frequency Division Multiple Access*)). TSCH utilise en plus un algorithme de saut de fréquences, conformément auquel les trames sont envoyées sur l'un ou sur l'autre des canaux en fonction du créneau alloué [4]. Les nœuds maintiennent une synchronisation précise en temps en compensant la dérive de leur horloge interne à chaque transmission. Cette approche permet aux réseaux à faible consommation (LLN) d'être plus robustes (résistance aux évanouissements de canal) ; elle leur permet également d'être déterministes au niveau de l'émission des trames, ce qui est particulièrement utile lorsqu'une certaine qualité de service doit être garantie ; elle leur permet également d'être en grande partie déterministes au niveau de la consommation d'énergie, un gros avantage pour les nœuds qui récoltent l'énergie dans leur environnement.

Adressage Selon la norme, chaque appareil doit avoir une adresse unique dans le réseau (notée EUI (*Extended Unique Identifier*), ou EUI-64 du fait qu'elle soit sur 64 bits). Celle-ci sert pour distinguer les nœuds entre eux hors contexte, et lors de la procédure d'association au réseau. Une fois associés au réseau, ils déterminent une adresse courte (sur seulement 16 bits) pouvant être utilisée lors de l'émission des trames — l'utilisation d'adresses courtes fait gagner 12 octets par trame, soit environ 10 % de sa taille maximale.

IEEE 802.15.4 définit l'utilisation d'une adresse de multidiffusion (*broadcast*) afin de transmettre des trames à tous les appareils dans leur environnement. Une trame multidiffusée consomme plus de ressources qu'une trame monodiffusée (*unicast*), que ce soit au sujet de l'énergie consommée (c'est le cas p. ex. de la couche MAC ContikiMAC [90], où une trame multidiffusée est retransmise en boucle pendant tout un cycle d'écoute), au sujet de l'utilisation de la bande passante (même exemple) ou d'instants d'écoute précis (TSCH p. ex. réserve des créneaux précis pour la transmission de trames multidiffusion). De plus, étant ciblés par l'adresse du paquet en couche physique, les nœuds sont contraints de recevoir l'intégralité du paquet avant de déterminer s'il leur est réellement destiné dans les couches supérieures, plutôt que de filtrer déjà à

4. TSCH ressemble ainsi à la technique d'étalement de spectre FHSS (*Frequency Hopping Spread Spectrum*), où le signal lui-même fait des sauts de fréquence ; mais l'« étalement » a lieu ici d'une trame à l'autre au lieu d'être à l'intérieur même d'une trame au niveau du signal transmis — à la couche MAC plutôt qu'à la couche physique.

la couche MAC et juste après la réception de l'adresse. En pratique, ces trames sont surtout utilisées dans les phases de découverte et de routage. Quelques mécanismes de découverte de service peuvent exister, comme par exemple le service *multicast DNS* [33]. Aussi, les trames multidiffusées ne sont pas acquittées.

1.2.3 Couches supérieures et implémentations

La norme IEEE 802.15.4 définit uniquement la 1re et une partie de la 2e du modèle OSI. Les couches supérieures jusqu'à la couche application (la 7e) sont définies par d'autres spécifications.

6LoWPAN, couche d'adaptation à IPv6 6LoWPAN (*IPv6 over Low-Power Wireless Personal Area Networks*), décrit par la RFC 4944 [25], est une initiative de l'IETF (*Internet Engineering Task Force*), dont le rôle est de développer et de promouvoir des normes internet, et entre autres la suite de protocoles IP (*Internet Protocol*). Elle se glisse entre la couche liaison de données et la couche réseau IPv6 (*Internet Protocol, version 6*) (cf. figure 1.3). Son but est d'interconnecter les capteurs des réseaux personnels sans fil à faible puissance aux réseaux IPv6, et par là à tout internet. D'après la RFC (*Requests For Comments*) d'IPv6 :

> IPv6 exige que chaque lien dans l'internet ait un MTU (*Maximum Transmission Unit*) d'au moins 1280 octets. Sur tout lien qui ne peut acheminer un paquet de 1280 octets en un seul morceau, la fragmentation et le réassemblage spécifiques au lien doivent être fournis à une couche inférieure à IPv6.
>
> — RFC 2460 [17]

Le MTU de IEEE 802.15.4 est de 127 octets de PDU à la couche physique (soit une durée de 4 ms de transmission pour une trame complète à 250 kbit/s), donc l'une des tâches de 6LoWPAN est d'implémenter cette couche de fragmentation.

En plus de la conformité stricte à IPv6, 6LoWPAN propose aussi des adaptations de la norme IPv6 complète aux contraintes des réseaux de capteurs sans fil. Un des points essentiels qu'il propose est la compression d'en-tête. Lorsqu'un paquet IPv6 standard est envoyé, il occupe déjà la majeure partie de la place disponible dans la trame IEEE 802.15.4 : les en-têtes des couches liaison de données (minimum 25 octets dont 2 de FCS), IPv6 (minimum 40 octets) et UDP (*User Datagram Protocol*) (8 octets) occupent au total au minimum 73 octets, et cela sans compter l'en-tête de sécurisation de la couche liaison de données (21 octets), fortement recommandée. Il ne reste au final que 33 octets pour les données elles-mêmes, ce qui implique une fragmentation fréquente. C'est encore pire lors de l'utilisation de TCP (*Transport Control Protocol*), où il ne reste plus que 21 octets pour les données.

Pour libérer un maximum d'espace pour les données, 6LoWPAN propose de compresser les en-têtes. Le résultat final escompté par l'IETF est que « de petites charges utiles d'application, associées à une compression d'en-tête appropriée, produiront des paquets qui rentrent dans une seule trame IEEE 802.15.4. » [25]. La compression d'IPv6 et d'UDP est proposée sans contexte dans la RFC originelle [25] [5]. Certains champs, pouvant être déduits d'autres, sont omis (p. ex. le préfixe de réseau IPv6, c'est-à-dire la moitié haute des adresses IPv6 locales, connues par tous les nœuds, ou aussi le champ *length* d'UDP pouvant être déduit de celui d'IPv6 et celui d'IPv6 pouvant être déduit de la couche liaison de données) ; d'autres voient leur taille réduite pour répondre efficacement aux cas les plus fréquents (p. ex. le numéro de port UDP réduit à 4 bits lorsqu'il est dans la plage 61 616–61 631, c'est-à-dire en hexadécimal F0B0–F0BF) ; d'autres enfin, ne correspondant pas à un cas standard, sont transmis tels quels (p. ex. le champ IPv6 *hop-limit*). Une compression

5. En fait, le seul contexte partagé est l'appartenance au réseau 6LoWPAN, que tout capteur joignant ce réseau possède de toute façon. Aucun contexte n'est conservé au sujet d'une connexion.

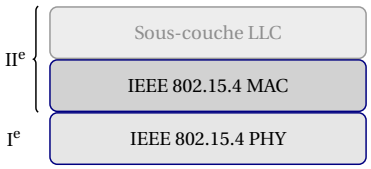


FIGURE 1.2 – Pile de protocoles de la norme IEEE 802.15.4.

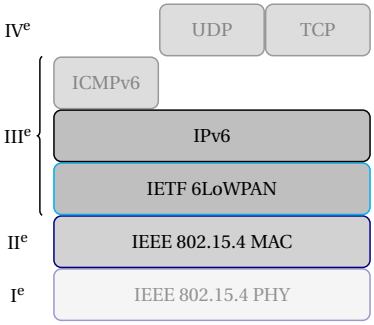


FIGURE 1.3 – Pile de protocoles de 6LoWPAN.

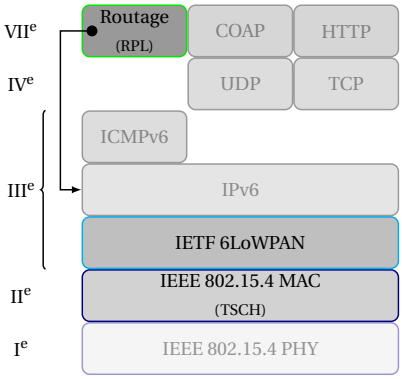


FIGURE 1.4 – Pile de protocoles de 6TiSCH.

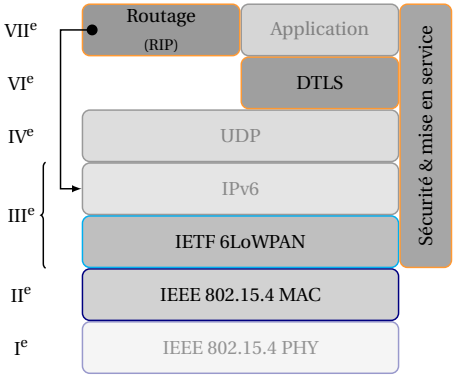


FIGURE 1.5 – Pile de protocoles de Thread.

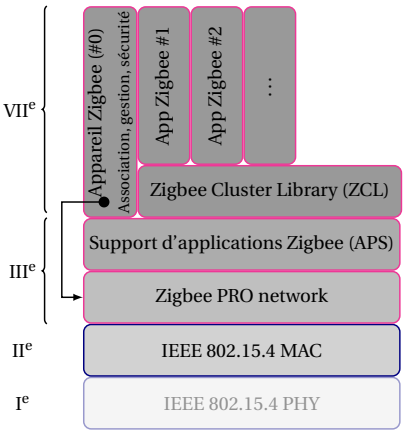


FIGURE 1.6 – Pile de protocoles de Zigbee 3.0.

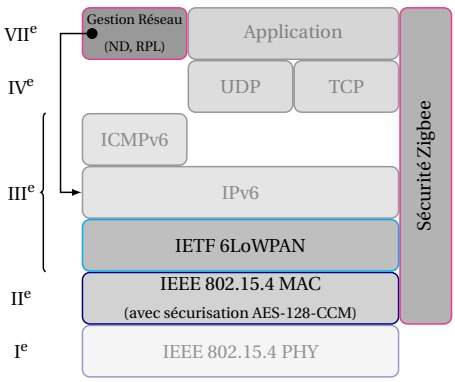
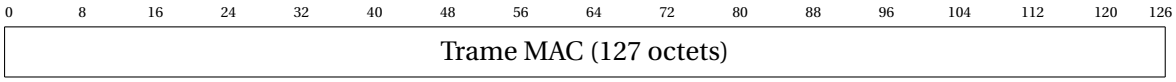


FIGURE 1.7 – Pile de protocoles de Zigbee IP.



(a) PDU d'IEEE 802.15.4 en couche physique (taille maximale)

(b) PDU d'IEEE 802.15.4 en couche MAC transportant un paquet IPv6, avec en-tête de sécurisation

(c) Paquet UDP sur IPv6 dans une trame IEEE 802.15.4 (sans compression)

(d) Paquet TCP sur IPv6 dans une trame IEEE 802.15.4 (sans compression)

(e) Paquet UDP sur IPv6 dans une trame IEEE 802.15.4, avec compression 6LoWPAN des en-têtes IPv6 et UDP.

FIGURE 1.8 – Comparaison de la place occupée par les différents en-têtes dans une trame IEEE 802.15.4.

plus poussée avec un contexte partagé est décrit dans un autre document [31]. La compression des en-têtes TCP [45] et ICMPv6 [46] sont décrites par des *drafts* mais pas encore standardisés.

6TISCH, connexion IPv6 par-dessus le mode TSCH L'architecture 6TiSCH (*IPv6 over the TSCH mode of IEEE 802.15.4e*) [37], produit par le groupe de travail éponyme de l'IETF [122], étend les fonctionnalités offertes par le mode de transmission TSCH (allocation de cellules temporelles & fréquentielles pour offrir un réseau robuste et déterministe) au-delà de la couche liaison de données. Elle décrit l'allocation statique ou dynamique des cellules de transmission (en temps et en fréquence) à l'aide du protocole 6top (*6TiSCH Operation Sublayer*) [38]. Comme le montre la pile protocolaire en figure 1.4, il s'appuie sur 6LoWPAN et IPv6 comme couche réseau, et au-delà sur les protocoles de la couche transport UDP et TCP ainsi que de la couche application, CoAP (*Constrained Application Protocol*) et HTTP (*HyperText Transfert Protocol*) — protocoles communs sur internet, et facilement interopérables avec d'autres utilisateurs.

OpenWSN (*Open Wireless Sensor Networks*) [124] est une projet à code source ouvert de l'université de Berkeley. Il s'agit d'un système d'exploitation pour réseaux de capteurs, qui se veut conforme à l'architecture 6TiSCH et aux normes en place. La gestion de la structure d'arbre sous-jacente à TSCH ainsi que des transmissions multisauts est faite à l'aide du protocole de routage RPL (*Routing Protocol for LLN*) [32] en mode *non-storing*.

Zigbee La *Zigbee Alliance* est une association d'entreprises à but non lucratif fondée en 2002, et qui compte maintenant plus de 500 membres. Elle définit un ensemble de protocoles de communications haut niveau pour la création de WPAN (*Wireless Personnal Area Networks*), nommé Zigbee. Le modèle de contraintes pris en compte est celui des réseaux de capteurs sans fil, avec des radios à bas coût, à faible puissance et à faible débit — il s'appuie ainsi sur les couches physique et MAC de IEEE 802.15.4. Le mode de communications utilisé est le mode sans balises (*nonbeacon-enabled*) ou le mode synchronisé avec balises (*beacon-enabled*), les deux modes supportés le plus tôt dans les différentes versions de la norme IEEE 802.15.4 (depuis 2003). La sécurisation des communications est prise en compte par la structure, en activant le chiffrement basé sur AES-128 dès

la couche liaison de données. Le réseau créé est flexible et extensible, autoconfigurant et autoréparant.

La Zigbee Alliance définit différentes normes. Zigbee 3.0 [135] est la norme historiquement la plus présente. La pile de protocole, présentée en figure 1.6, montre qu'elle repose entre autres sur ses propres protocoles de communication (comme la couche réseau *Zigbee PRO network*, qui utilise AODV (*Ad-hoc On-demand Distance Vector*) comme protocole de routage). Elle définit également une couche applicative commune, la ZCL (*Zigbee Cluster Library*), où les objets sont décrits avec leurs attributs et leurs fonctionnalités à travers des commandes applicatives normalisées pour un groupe donné (appelé grappe, ou *cluster*). Toutefois, elle ne définit pas de mécanisme d'économie d'énergie, c'est-à-dire pas de ratio d'activité cyclique.

La Zigbee Alliance tend toutefois aujourd'hui à se rapprocher de l'interopérabilité avec d'autres technologies. La norme Zigbee IP, développée en parallèle de Zigbee 3.0, s'est concentrée sur l'utilisation des normes de l'IETF (IPv6, ICMPv6, UDP, TCP… cf. figure 1.7). D'un autre côté, la ZCL a été renommée en janvier 2017 en Dotdot [134], et est compatible avec d'autres piles réseau, entre autres avec le WiFi (IEEE 802.11), le Bluetooth, ou encore Thread.

Thread Thread est un protocole réseau sans fil et maillé, développé par le *Thread Group* [133]. La figure 1.5 montre la pile protocolaire qu'il utilise. Elle est basée sur IEEE 802.15.4 comme couches physique et liaison de données, ainsi que sur les normes internet de l'IETF (IPv6, 6LoWPAN, UDP). Il est présenté comme étant sans point unique de défaillance (mis à part peut-être le routeur de bordure, lorsqu'il n'y en a qu'un). Plusieurs implémentations existent, dont l'implémentation à code source ouvert OpenThread, développé par Nest Labs et par Google [132].

Thread fonctionne à l'aide d'un coordinateur central, appelé le *leader*, qui choisit les fréquences utilisées et diverses autres informations de sécurité. Ensuite, une série de nœuds, appelés *routeurs*, forment une dorsale de nœuds, toujours actifs, et utilisent le protocole de routage RIP (*Routing Information Protocol*) [18] pour la communication multisaut. Leur objectif est de former un réseau toujours disponible pour les nœuds périphériques, qui peuvent alors économiser massivement l'énergie en éteignant leur radio. La durée de vie de ces nœuds périphériques peut alors atteindre la dizaine d'années quand bien même ils ne seraient alimentés que par de petites batteries [133].

Au niveau de la couche transport, Thread utilise UDP, DTLS (*Datagram Transport Layer Security*) et CoAP [35]. La couche applicative n'est pas spécifiée, Thread étant essentiellement une pile protocolaire de communication. N'importe quelle couche application peut être supportée tant qu'elle est à faible débit et opérable par-dessus IPv6 [133]. Nous avons vu l'exemple de Dotdot, mais d'autres peuvent également y être déployées comme l'environnement libre IoTivity de la fondation OpenConnectivity [131].

Contiki Contiki-OS [66] est un système d'exploitation léger, initialement créé par le centre suédois de recherche scientifique SICS. Son développement a été poursuivi jusqu'à la version 3.0, puis a été repris depuis 2017 par un *fork*, nommé Contiki-NG [125]. L'objectif de cette reprise est entre autres la modernisation de la structure de développement, et un support de plateformes modernes.

Contiki est spécialement conçu pour l'exploitation pour les microcontrôleurs et capteurs sans fil à basse consommation, en favorisant une consommation énergétique et une empreinte mémoire minimale. Il offre un environnement de développement facilité pour créer des applications sur capteurs agnostique de la plateforme de travail, et compatibles entre eux puisqu'ils sont appuyés sur les standards IEEE et IETF. Il supporte la couche MAC IEEE 802.15.4 en mode sans balises (*non-beacon*) et en mode TSCH, et la couche Bluetooth Low-Energy (support expérimental). La couche réseau qu'il utilise est 6LoWPAN/IPv6, et propose l'utilisation du protocole de routage

RPL. L'application est définie par le développeur qui utilise le système d'exploitation, qui propose néanmoins une implémentation des protocoles les plus utilisés dans ce contexte, comme CoAP (et UDP), ou un service DNS (*Domain Name System*).

1.3 Le travail de routage

La portée de la 2e couche du modèle OSI (couche liaison de données) est limitée au lien physique, c'est-à-dire l'ensemble des appareils avec lesquels la communication est possible sans intermédiaires. La portée de la 3e couche (couche réseau) est « infinie » — tout appareil compatible avec le protocole réseau utilisé (en général IP), modulo quelques restrictions d'accès à certaines zones p. ex. à cause d'un filtrage par un pare-feu ou bien de la portée limitée d'une adresse privée. Le travail de routage correspond à l'acheminement des messages à travers un réseau multi-saut jusqu'à une destination donnée, chaque nœud devant déterminer quel sera le prochain saut (*next-hop*), qui doit être nœud voisin accessible à la couche liaison de données, à qui le message doit être transmis. Lors de la transmission d'un message, un nœud consulte sa table de routage, qui désigne le prochain saut à utiliser pour joindre la destination indiquée dans le message — tout comme un panneau indicateur donnant la direction des différentes villes qu'on peut atteindre en prenant l'une ou l'autre des sorties d'un carrefour. Le rôle du protocole de routage est d'organiser l'exploration de la topologie du réseau et de maintenir à jour cette table de routage, non seulement *sur* un nœud, mais également *entre* les nœuds — il s'agit d'un travail distribué entre les nœuds du réseau. Vues l'une après l'autre, les tables de routage décrivent ainsi des routes le long desquelles les messages pourront être aiguillés.

En plus que le travail normal de routage dans un réseau standard, le routage en réseaux de capteurs sans fil doit prendre en compte deux contraintes supplémentaires. Tout d'abord, les capteurs sont des plateformes avec des ressources limitées, tant au niveau mémoire qu'au niveau de la puissance de calcul ou de l'énergie disponible. Ensuite, le mode de communication est sans fil, c'est-à-dire un canal partagé, non seulement entre les capteurs eux-mêmes, mais aussi entre les technologies (cf. section 1.2.1). Les caractéristiques de transmission des liaisons radio peuvent beaucoup varier, soit à cause de mouvements d'objets dans l'environnement, soit à cause de la mobilité des capteurs, ce qui est beaucoup moins le cas dans des fibres optiques, très bien isolées des interférences extérieures, ou même des faisceaux hertziens, fixes et directifs.

1.3.1 Boucles de routage

Une boucle de routage est une situation où, suite à une désynchronisation entre les tables de routage des nœuds du réseau, plusieurs d'entre eux s'envoient successivement un paquet les uns aux autres. Cette situation peut arriver entre deux nœuds, mais peut aussi en impliquer un beaucoup plus grand nombre.

Ces boucles de routage sont bien sûr indésirables, et ce pour deux principales raisons. Tout d'abord, un paquet de données envoyé suivant cette boucle n'arrivera jamais à sa destination. Le service ne sera donc pas fourni pour ce paquet. La plupart des réseaux sont certes tolérants aux pertes, mais ces pertes perdureront pour tous les autres paquets empruntant la boucle tant que celle-ci continuera d'exister, interrompant tous les flux impactés et interrompant ainsi l'intégralité du service. De plus, le paquet ne s'arrêtera pas après un tour complet, mais effectuera plusieurs tours dans la boucle, jusqu'à expiration de sa durée de vie (*time to live* dans IPv4 et *hop limit* dans IPv6), impliquant une occupation du réseau [6] et une consommation énergétique supplémentaire,

6. Dans certains cas, la suroccupation du réseau peut aller jusqu'à saturer le réseau et perturber le trafic de contrôle, retardant d'autant la réparation de la boucle.

désirable nulle part, et encore moins dans les réseaux de capteurs sans fil où l'énergie et la bande passante sont des ressources limitées.

Le traitement des boucles de routage se fait différemment en fonction du protocole, suivant la conception du protocole et l'information que chaque nœud possède, et est fait de façon plus ou moins absolue. Certains tolèrent des boucles temporaires mais non pas stables (p. ex. RIP), d'autres démontrent leur absence complète (p. ex. BGP). La section suivante décrit entre autres les réactions de quelques protocoles de routage à ces boucles, suivant leur conception.

1.3.2 Classes de protocoles de routage

Les protocoles de routage travaillent sur un graphe reflétant la topologie du réseau. L'objectif des protocoles de routage étant de trouver le plus court chemin sur ce graphe entre deux nœuds. Pour réaliser ce calcul, une partie d'entre eux utilisent **l'algorithme de Bellman-Ford distribué**. Cet algorithme a l'avantage d'être facilement distribuable sur les nœuds du réseau : chaque nœud partage à ses voisins un **vecteur de distances** vers les autres nœuds du réseau. La diffusion des routes se fait ainsi de proche en proche. Cet algorithme est particulièrement intéressant pour les réseaux de capteurs sans fil, parce qu'il limite les ressources mémoire exigées pour chaque nœud : aucun des nœuds n'a besoin d'avoir une vision globale du réseau, mais seulement des prochains sauts (*next-hops*) à utiliser pour atteindre une destination. Un de ses inconvénients est le comptage à l'infini, une situation où, suite à la perte d'un lien, deux ou plusieurs nœuds entre dans une boucle de routage où ils s'utilisent successivement comme prochain saut, s'éloignant les uns après les autres de la destination. Tant que les nœuds sont en train de compter à l'infini, leur différentes tables de routage impliquent une boucle de routage. Les protocoles de routage résolvent ce scénario de façon différente. RIPv2 [18], par exemple, laisse les nœuds temporairement dans la boucle jusqu'à ce que les nœuds soient à une distance de 16 de la destination — 16 étant la plus grande valeur possible, et signifiant aussi que la destination est inaccessible. RPL [32] utilise un peu la même approche, mais définit la limite maximale en fonction de la plus proche position que le nœud a eu de la destination. D'autres protocoles de routage comme DSDV (*Destination-Sequenced Distance Vector*) [58] et plus tard AODV [62, 22] ou Babel [28] utilisent un **numéro de séquence** pour dater chronologiquement l'information connue sur une route, et évitent ainsi aux nœuds d'écraser une information fraîche par une autre ancienne. BGP-4 [23], quant à lui, utilise une variante de vecteurs de distances, appelée **vecteur de chemins**, où les nœuds joignent au préfixe annoncé l'ensemble des AS (*Autonomous System*) traversés (qui constitue un *AS Path*) jusqu'à la destination, permettant entre autres d'éviter les boucles entre les AS.

D'autres protocoles sortent de cette famille de protocoles à vecteur de distances. Appelés protocoles à états de lien, ils calculent l'ensemble des plus courts chemins avec l'algorithme de Dijkstra. Cette méthode requiert que chaque nœud connaisse la topologie exacte du réseau dans son ensemble, c'est-à-dire l'état de tous les liens du réseau — d'où leur nom. Un exemple très connu est celui d'OSPF (*Open Shortest Path First*) [19], prévu pour remplacer RIP et servir de protocole de routage à l'intérieur des AS [7]. Les boucles de routage sont gérées de façon plus simple que pour un routage à vecteur de distances, parce que les nœuds connaissent l'ensemble du réseau ; mais il faut cependant que le protocole maintienne la représentation de cette topologie cohérente entre les nœuds, ce qui peut impliquer une grosse charge en communication de contrôle. Cette raison, ainsi que le fait que les contraintes au niveau de la mémoire et des capacités de calculs augmentés, fait que ces protocoles sont moins adaptés aux LLN (*Low-Power and Lossy Networks*) et aux réseaux de capteurs sans fil. Il faut néanmoins citer des propositions de Cisco Systems [41, 43] et de OGIER et SPAGNOLO [42] sollicitées par l'IETF visant à étendre OSPFv3 pour qu'IPv6 puisse

7. C'est en tout cas la recommandation de l'IESG (*Internet Engineering Steering Group*) à travers la RFC 1371 [14].

fonctionner efficacement dans les environnements MANET (*Mobile Ad Hoc Networks*) [74]. Le protocole de routage OLSR (*Optimized Link State Routing Protocol*) [21, 34] est lui aussi à états de lien et optimisé pour les MANET. OLSR permet entre autres de réduire le nombre de relais, appelés MPR (*Multi-Points Relays*), utilisés pour la diffusion dans le réseau.

Ces deux approches, vecteur de distances et états de lien, donnent deux familles principales de protocoles de routage. ZRP (*Zone Routing Protocol*) [60, 40] cherche à combiner les avantages de l'un et de l'autre, en définissant des zones à l'intérieur desquelles le routage est à états de lien, et à l'extérieur desquelles il est à vecteur de distances. Cette approche hybride permet d'avoir un très bonne connaissance du réseau avec un horizon proche, et de pouvoir toujours atteindre les nœuds éloignés.

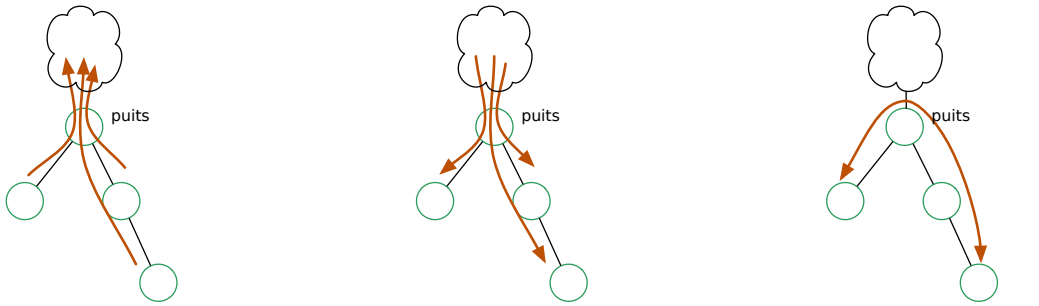
1.3.3 Arbres de collecte

Les réseaux ad hoc comme les MANET se construisent de façon décentralisée et sans infrastructure préexistante. Les nœuds s'organisent de façon autonome, et retransmettent les données des voisins en fonction de la topologie du réseau et de l'algorithme de routage. Lorsqu'il n'y a pas de hiérarchie entre les nœuds, le réseau est dit maillé (*mesh network*).

Partant de l'hypothèse que les réseaux de capteurs sont principalement des producteurs de données, l'information est disséminée à l'intérieur du réseau et doit être rapatriée vers les routeurs de bordure, points de sortie du réseau. L'essentiel du trafic est ainsi « ***multipoint-to-point*** », c'est-à-dire qu'il converge vers un des points du réseau appelé le puits (*sink* en anglais) en provenance de n'importe quel hôte du réseau (cf. illustration de la figure 1.9a). Pour gérer efficacement ce type de trafic, une des techniques les plus utilisées est d'organiser le réseau sous forme d'arbre de collecte, un graphe orienté sans boucle, où tous les arcs, correspondant à des **routes par défaut**, pointent en direction du routeur de bordure. L'arbre de collecte est appelé un DODAG (*Destination-Oriented Directed Acyclic Graph*) lorsque les nœuds enregistrent plusieurs routes par défaut alternatives. Le protocole de routage est responsable de l'établissement de cet arbre de collecte, mais également de sa maintenance en réponse aux changements de la topologie du réseau. L'arbre peut être reconstruit à partir de zéro dans ces situations ; le concept de *réparation locale* vient toutefois avec l'idée de ne pas solliciter tout le réseau, mais uniquement une sous-partie — ce que TORA (*Temporally-Ordered Routing Algorithm*) [61, 39] propose de faire pour toutes les routes.

Même s'il est envisageable de faire fonctionner un réseau producteur de données avec uniquement un arbre de collecte des données, ce n'est dans la plupart des cas pas suffisant. En effet, l'accès à un nœud depuis l'extérieur doit être possible, pour des raisons par exemple de configuration, d'acquittement de couche transport ou application, ou d'échange de clefs de chiffrement. Il s'agit du trafic « ***point-to-multipoint*** », qui, partant du routeur de bordure, est distribué à n'importe quel hôte du réseau (cf. illustration de la figure 1.9b). Le routage de ce trafic est réalisé par l'établissement de **routes d'hôtes** (des routes pointant vers un hôte précis du réseau, appelées aussi routes descendantes), qui doivent alors être établies et maintenues séparément pour chaque capteur. La plupart des protocoles installent la route sur les nœuds du réseau, comme n'importe quelle autre route. D'autres, à l'instar de DSR (*Dynamic Source Routing*) [57, 24] utilisent la technique du routage à la source, où la source indique dans l'en-tête du paquet à transmettre le chemin qu'il doit suivre. RPL [32] propose une fonctionnalité identique, nommée *Non-storing mode*, dans laquelle les nœuds intermédiaires n'ont pas besoin d'installer la route d'hôte dans leur table de routage.

Le trafic de type « ***point-to-point*** », où les paquets sont routés entre deux nœuds quelconques du réseau (cf. illustration de la figure 1.9c), est naturellement pris en charge en utilisant une structure d'arbre. En effet, le trafic remonte naturellement l'arbre de collecte en suivant les routes par



(a) Trafic « *multipoint-to-point* » (b) Trafic « *point-to-multipoint* » (c) Trafic « *point-to-point* »

FIGURE 1.9 – Trois types de flux de trafic

défaut avant de redescendre le long de l'arbre en suivant une route d'hôte. La route empruntée est effectivement inefficace, puisque le trafic doit passer par le centre du réseau avant de rejoindre sa destination ; toutefois, ce type de trafic est peu attendu en réseau de capteurs sans fil, et l'optimalité de son acheminement est moins importante que pour les autres types de trafic.

La communauté MANET s'est jusque-là plus concentrée sur la construction et la maintenance des routes par défaut et d'hôtes dans un réseau plat que sur l'injection de préfixes, bien que OLSR [21] permette de publier des HNA (*Host and Network Associations*). L'hypothèse tacite est que les adresses sont auto-affectées et par là réparties aléatoirement dans le réseau, ce qui empêche l'agrégation des routes.

1.3.4 Proactif *versus* réactif

L'instant où une route est calculée est un choix entre deux possibilités : soit travailler en avance en précalculant les routes dont on a besoin (c'est-à-dire être *proactif*), soit attendre d'avoir besoin d'une route pour la calculer (c'est-à-dire être *réactif*). Il n'y a pas de choix meilleur que l'autre sur tous les plans, le meilleur choix dépend des caractéristiques du système déployé, de l'environnement, et des objectifs fixés. D'un côté, être proactif permet d'avoir une information toujours prête et à jour ; cela implique pourtant aussi un coût en ressources (énergie de transmissions et occupation du canal pour les messages de contrôle, calculs, stockage mémoire…) constant même lorsque les routes calculées ne sont en fait pas utilisées. De l'autre côté, être réactif permet de limiter les échanges de contrôle à des instants et pour des requêtes précises ; toutefois, cela implique un temps de latence et un coût en ressources pour établir l'information au moment où elle doit être utilisée. La proactivité sera utilisée par exemple dans des cas où une faible latence doit être garantie, ou dans le cas de réseaux déterministes — le cas typique de 6TiSCH [37], qui utilise RPL comme protocole de routage. La réactivité sera plutôt utilisée dans des situations où les cibles sont a priori inconnues, où l'état du réseau change très rapidement ou au contraire lorsque le réseau est très stable et que les nœuds doivent rester silencieux.

Dans les réseaux de capteurs, à ce sujet, deux protocoles peuvent être cités en exemple. RPL est principalement proactif. L'arbre de collecte est calculé au départ et maintenu à jour grâce à algorithme *Trickle* [30]. Les routes d'hôtes sont également maintenues régulièrement. Il en résulte naturellement que RPL n'est jamais silencieux. AODV, de son côté, est réactif. Les routes sont découvertes uniquement lorsque cela est nécessaire, ce qui le rend entièrement silencieux lorsqu'aucune nouvelle information n'est requise. Cela implique par contre des inondations fréquentes du réseau lorsqu'un hôte y est recherché. Chacun de ces protocoles sont adaptés à des conditions particulières. VUČINIĆ, TOURANCHEAU et DUDA [98] avancent par exemple que RPL est plus adapté dans le cas spécifique d'un scénario de domotique que LOADng (*Lightweight On-demand Ad-hoc*

Distance-vector Routing Protocol — next generation) (un protocole proche de AODV), compte tenu du trafic de données proposé avec leurs expérimentations. YI, CLAUSEN et IGARASHI [99] reportent des faiblesses de RPL pour la gestion des routes descendantes justement à cause de sa proactivité lors de la construction de routes d'hôtes, là où la structure même de LOADng l'avantage, puisqu'il n'y a pas d'arbre de collecte donc pas de routes descendantes.

1.3.5 Caractérisation des protocoles de routage

Le tableau 1.2 donne une vue générale des principales caractéristiques dont les sections précédentes parlent, décrites ci-dessus pour quelques protocoles de routage. Les protocoles « classiques » sont tout d'abord présentés, avec RIP, sûrement le protocole de routage le plus ancien en version 1 et toujours utilisé dans sa version 2 pour le routage à l'intérieur des AS, mais aussi OSPF et IS-IS (*Intermediate System to Intermediate System*), ses deux grands successeurs, et BGP (*Border Gateway Protocol*) conçu pour le routage entre les AS. GARCIA-LUNES-ACEVES [56], s'appuyant sur l'idée des calculs diffusants de DIJKSTRA et SCHOLTEN [54], a proposé DUAL (*Diffusing Update Algorithm*), une famille d'algorithmes de mise à jour qui pousse l'information dans le sous-arbre concerné, et qui est utilisé par EIGRP (*Cisco's Enhanced Interior Gateway Routing Protocol*) et Babel. OLSR utilise l'approche à états de lien dans les réseaux mobiles, et sélectionne pour ça des nœuds spéciaux (MPR) pour diminuer le nombre de messages émis lors d'une inondation et le nombre de liens à prendre en compte dans les calculs ; de son côté, BATMAN (*better approach to mobile ad hoc networking*) propose une approche opportuniste pour le même problème, en se basant sur des nœuds centraux et des liens fiables connus par les nœuds. ZRP combine l'approche à vecteur de distances pour les destinations éloignées et à états de lien pour les destinations proches. DSR pose les bases du routage à la source pour les MANET et DSDV celles du routage à vecteur de distances séquencées pour combattre les boucles de routage. Lorsque la destination n'est plus joignable par une partie du réseau suite à un changement de topologie, TORA ne vise pas à rétablir des plus court chemin, mais présente plutôt une technique de réparation locale des routes où les messages de contrôle ont une portée réduite et où une petite partir du réseau seule est impactée. AODV et LOADng proposent une approche entièrement réactive pour la gestion du routage dans ces réseaux. Enfin, RPL s'occupe plus spécifiquement du routage pour les réseaux de capteurs sans fil où la puissance des nœuds est petite et les coûts de communication élevés.

Ces trois derniers protocoles se rapprochent le plus des objectifs que fixés dans l'introduction. Nous verrons dans la section 2.2 leurs points forts et leurs points faibles, et pourquoi j'ai choisi de proposer une approche différente en vue d'obtenir un résultat répondant mieux à la problématique de routage dans réseaux de capteurs sans fil.

TABLE 1.2 – Caractérisation de quelques protocoles de routage.

Protocole	Date	Conçu pour	Classe	Proactif / réactif	Notes
RIP [12]	1988	Internet (IGP)	DV	Proactif	Un des premiers protocoles de routage (en v1)
— (v2) [18]	1998				
BGP (v4) [15, 23]	1995	Internet (EGP)	PV	Proactif	
OSPF (v2) [19]	1998	Internet (IGP)	LS	Proactif	
— (pour IPv6) [20]	1999				
IS-IS [13, 137]	2002	Internet (IGP)	LS	Proactif	
Babel [28]	2011	Avec & sans fil	SDV	Proactif	Conçu pour les réseaux maillés comme pour les réseaux basés sur des préfixes, avec et sans fil
EIGRP [36]	2016	Réseau maillé	DV (LS)[a]	Proactif	Utilisation de DUAL
OLSR [21]	2003	MANET	LS	Proactif	
— (v2) [34]	2014				
BATMAN [128]					
— (draft) [44]	2008	MANET	DV	Proactf	
DSR [57]	1994	MANET	LS	Réactif	Routage à la source
— (RFC) [24]	2007				
TORA [61]	1997	MANET	LR	—[b]	Portée réduite des messages de contrôle lors d'un changement de topologie
— (draft) [39]	2001				
ZRP [60]	1997	MANET	LS / PV[c]	Les deux[c]	Séparation du réseau en zones, avec des règles de routage différentes
— (draft) [40]	2002				
DSDV [58]	1994	MANET	SDV	Proactif	Utilisation des numéros de séquence pour éviter les boucles
AODV [62]	1999	MANET	SDV	Réactif	Routage à la demande en réseaux de capteurs sans fil
— (RFC) [22]	2003				
LOADng [92, 111]	2012	MANET	SDV	Réactif	Variante d'AODV, avec état interne réduit et réponses RREP depuis la destination
— (draft) [49]	2016				
RPL [32]	2012	LLN	SDV	Proactif	

Classes : — DV : Protocole à vecteur de distances — LS : Protocole à états de lien — PV : Protocole à vecteur de chemins
— SDV : Protocole à vecteur de distances séquencées — LR : Protocole à reversement des liens

[a] EIGRP est DV, mais il envoie aussi des mises à jour à états de lien lorsque ceux-ci changent.

[b] TORA ne définit pas du tout les instants où une route est calculée, mais seulement la façon dont elle est calculée. Il s'annonce lui-même comme pouvant « simultanément prendre en charge le routage à la demande initié par la source [...] et le routage proactif initié par la destination [...]. » [39]

[c] Le protocole de routage intrazone est LS et proactif ; le protocole de routage interzones est PV et réactif.

1.4 Mesurer le coût d'un lien

Pour atteindre une destination donnée et établir une route jusqu'à elle, l'algorithme de routage a souvent le choix entre plusieurs chemins, c'est-à-dire un enchaînement de liens à traverser. Le choix d'un chemin ou d'un autre est fait en fonction de plusieurs critères, plus ou moins importants suivant les objectifs fixés : le coût d'établissement et de maintenance des routes (stabilité des routes, impacts des messages de contrôle circulant eux-mêmes dans le réseau), le coût de l'utilisation des routes en proportion de la quantité d'information utile envoyée (nombre de retransmissions, énergie dépensée, occupation du canal de communication, facteur financier, équilibre de charge entre les nœuds…) et la qualité de service requise (latence, instabilité de la latence, débit offert, taux de pertes…). Afin de prendre les bonnes décisions de routage, il convient que l'algorithme de routage possède une description juste de la réalité physique des caractéristiques des liens, du coût de ce lien. Ce coût peut représenter l'énergie dépensée pour transmettre le paquet, le nombre d'émission, la bande passante occupée, le taux de perte, la latence constatée, ou n'importe quelle autre caractère du lien. Il doit être combinable, c'est-à-dire qu'on peut calculer le coût d'un chemin en fonction du coût des liens qui le composent. Il doit également être comparable via une relation d'ordre total indiquant lequel des deux chemins (ou liens) est préférable. Enfin, il doit être strictement positif (le coût négatif indiquerait qu'on *gagne* quelque chose en transmettant le paquet).

1.4.1 Quelques métriques courantes

Le problème de l'estimation de la qualité du lien est déjà bien traité dans la littérature. Les protocoles de routage peuvent prendre en compte des métriques de tous types, aussi arbitraires que pragmatiques. Un certain nombre d'entre elles sont spécialement conçues pour décrire l'état global d'un chemin, par exemple en fonction du nombre de nœuds qui l'utilisent ou du nombre de paquets qui y transitent, afin de faire de la répartition de charge [110]. Ces métriques sont intéressantes dans le sens qu'elles permettent d'éviter les situations où un chemin, offrant peut-être un accès légèrement plus facile jusqu'à la destination, se trouve être surutilisé par rapport aux autres. D'autres métriques encore décrivent les caractéristiques extrêmes le long du chemin, comme la plus petite bande passante des liens du chemin. Bien que très utiles à leur place, elles décrivent difficilement l'état d'un seul lien, ce qui est plus exactement ce qui nous intéresse ici, où nous cherchons à déterminer des caractéristiques décrivant un lien. Quelques-unes de celles-ci sont décrites ici.

La métrique hop count — littéralement **décompte des sauts** — est la plus naïve. Elle compte simplement les liens (ou sauts) traversés. Lorsque les liens présentent des qualités effectivement comparables, cette métrique est intéressante à cause de sa simplicité. La comparaison est triviale, le calcul du coût d'un chemin l'est tout autant, et il n'y a aucune problématique de mesure physique ou de portabilité. Cependant, dans le cas des réseaux de capteurs sans fil et de la communication sans fil en général, les liens longs seront privilégiés par cette métrique puisqu'ils amènent en moins de sauts jusqu'à la destination, alors que ce sont aussi les liens les plus faibles (le signal étant alors moins puissant). D'où la nécessité, en plus de leur nombre, de prendre en compte la qualité des liens.

Pour pallier aux erreurs de transmissions, la norme IEEE 802.15.4, comme d'autres normes, propose de demander un acquittement aux paquets transmis, pour garantir que le récepteur a reçu et déchiffré correctement le message. Lorsqu'aucun acquittement n'est reçu, le message est retransmis. Or, chaque transmission a le même coût pour le capteur : une certaine quantité d'énergie nécessaire pour l'émission du paquet. La métrique **ETX (*Expected Transmission count*)** cor-

respond au nombre statistique de retransmission jusqu'à ce que le récepteur ait acquitté le paquet [70]. Cette métrique est facile à manipuler : sa valeur est calculable en observant comment se sont déroulées les transmissions sur ce lien, ce qui est relativement précis sur des liens stables et utilisés, mais imprécise lors de la découverte d'un nouveau lien, sur des liens très peu utilisés ou qui présentent un comportement instable. L'envoi de trafic-sonde n'est pas idéal en réseau de capteurs sans fil, où chaque transmission coûte de l'énergie.

L'ETX est très proche du PER (*Packet Error Rate*), la probabilité que la trame ne soit pas correctement transmise : en prenant en compte l'acquittement et en supposant un nombre de retransmission infini, ETX = $\mathrm{PER_{paquet}}^{-1} \times \mathrm{PER_{acquittement}}^{-1}$. L'ETX est pourtant préféré en multisaut, parce que le coût d'un chemin est calculé comme la somme du nombre de transmissions sur chaque lien le long du chemin. Le PER perd de sa signification en multisaut parce que la retransmission en cas d'échec n'est pas faite de bout en bout, mais seulement sur le lien considéré.

Il est possible de combiner plusieurs métriques distinctes pour créer une métrique composée, avec ses propres règles de combinaison et de comparaison. Ainsi, la métrique composite usuelle d'EIGRP [36] utilise la bande passante et la somme des délais des liens et interfaces traversés, deux valeurs fixes pour une topologie donnée ; elle utilise aussi la charge et la fiabilité des liens, métriques explicitement décrites comme « [n'étant] pas mesurées dynamiquement ; elles ne sont mesurées qu'au moment où un lien change » — on remarque que cette métrique est définie pour bouger le moins possible, ce qui la rend inappropriée pour les réseaux sans fil où les liens sont soumis à toutes sortes de perturbations environnementales.

Ces métriques sont faciles à utiliser pour décrire le coût d'un chemin. D'autres indicateurs sont plus précis dans leur description de la qualité d'un lien précis, et peuvent ensuite servir de base à des métriques de routage plus complexes.

Lors de la réception d'un paquet de données, il est possible au récepteur de mesurer physiquement **la puissance radio** perçue sur l'antenne. Cette mesure peut ensuite être utilisée pour caractériser le coût d'un lien, les liens où la puissance du signal est faible étant ainsi évités. Cet indicateur est appelé RSSI (*Received Signal Strength Indicator*). Comme nous le verrons dans le chapitre 4, bien qu'efficace pour détecter les liens faibles, cet indicateur a un trop grand nombre de faux positifs : un lien sur lequel la puissance du signal est faible n'implique pas directement un lien de mauvaise qualité.

La norme IEEE 802.15.4 standard [10] suggère l'utilisation d'un autre indicateur, appelé **LQI (*Link Quality Indicator*)**, qui permettrait de caractériser la qualité du paquet reçu. Cet indicateur est également étudié dans le chapitre 4. Il possède le défaut que son calcul n'est pas clairement décrit, mais laissé au choix de l'implémentation, donc dépendante de l'implémentation, et peu portable d'un nœud à l'autre. Un autre point gênant est que, si il peut parfois bien décrire la qualité d'un paquet précis, il est difficile d'en extrapoler la qualité d'un lien en général.

Une mesure utilisée pour déterminer si le signal est démodulable est le rapport de la puissance du signal sur celle du bruit ambiant, noté **SNR (*Signal-Noise Ratio*)**. Dans le cadre théorique, cette mesure est très utile en ce qu'elle permet de déterminer, à l'aide de formules connues dépendantes de la modulation utilisée, la probabilité d'erreurs de transmission, c'est-à-dire le BER (*Bit Error Rate*). Son inconvénient est de rester trop théorique, et en pratique n'est pas mesurable par le module radio, qui est souvent incapable de distinguer entre l'énergie du signal et celle du bruit. De plus, elle pose les mêmes problèmes que le LQI en ce que, si elle décrit correctement la qualité du signal à un instant donné, elle n'est pas trivialement extrapolable à un lien. Nous verrons au chapitre 4 qu'il est en fait possible de l'exploiter en mesurant séparément la puissance du bruit et celle du signal.

1.4.2 Mesure bidirectionnelle

La qualité des transmissions radio dépend de l'environnement et du matériel utilisé. Les facteurs environnementaux clés sont le bruit ambiant, les obstacles physiques qui peuvent causer des réflexions ou des absorptions, et les interférences produites par des communications parallèles, potentiellement en provenance d'autres technologies. Le matériel radio impacte la transmission de par sa qualité et ses caractéristiques particulières. Ces effets jouent un rôle important dans les réseaux de capteurs sans fil, où les appareils sont bon marché et l'environnement peut être surchargé de transmissions. Mais ils peuvent en outre perturber distinctement l'un ou l'autre des nœuds et générer ainsi une asymétrie du lien, c'est-à-dire qu'il ne présente pas la même qualité dans les deux directions. C'est le cas lorsque des interférences perturbent la réception d'une des deux radios sans gêner l'autre. C'est également le cas lorsque les deux puces radio utilisées pour communiquer présentent un rapport gain-sensibilité différent ; on a alors une différence de budget de liaison entre les deux sens, et donc une asymétrie naturelle entre ces deux directions. Dans des cas extrêmes, il se pourrait même qu'un nœud soit « sourd » (c'est-à-dire que les autres nœuds reçoivent ses paquets, mais que lui ne reçoit rien en retour) ou inversement « muet » (c'est-à-dire que ce nœud reçoit convenablement, mais n'est pas entendu) par rapport à un ou plusieurs de ses voisins.

BRUN-LAGUNA et al. [109] rapportent, dans le cas d'un réseau déployé dans un verger de pêches, la quasi-absence d'asymétrie (grosso modo pas plus de 2 dB d'écart de puissance entre les deux sens du lien) ; toutefois, ces mesures sont effectuées dans un environnement précis, où les capteurs communiquent en ligne directe, sans interférences attendues, et utilisent la même plateforme pour la plupart des nœuds du réseau. Leurs conclusions sont donc à relativiser. D'autres travaux montrent eux que, bien que les liens soient généralement symétriques, l'asymétrie est bien présente [75, 85, 94].

Conclusion

La norme IEEE 802.15.4 fournit les bases essentielles pour la transmission de paquets sur des réseaux à bas débits, à faible puissance et à courte portée. Plusieurs couche physique, une couche MAC commune, et divers mode de transmission adaptés à l'économie d'énergie (*beacon-enabled*, TSCH, CSL) sont proposées, et leur implémentation et adaptation aux protocoles des couches supérieures sont déjà mise en œuvre, fournissant ainsi un ensemble de protocoles permettant de créer un réseau de nœuds communiquants. Au milieu de ce réseau où les transmissions doivent être faites en plusieurs sauts, un protocole de routage est nécessaire pour découvrir, établir et entretenir les routes nécessaires à l'acheminement des données dans le réseau. Son comportement doit être adapté aux réseaux de capteurs sans fil, c'est-à-dire à leurs liens variants, et à leurs nœuds légers contraintes en énergie. Pour répondre à cela, le routage doit être efficace, mais aussi garanti. C'est ce thème que nous abordons dans la première partie de cette thèse.

Ce travail de routage doit s'appuyer sur une connaissance de la topologie du réseau, afin que les décisions de routage reposent sur des bases solides. Pour cela, une estimation de la qualité des liens transformée en une métrique synthétique est nécessaire, décrivant les caractéristiques de transmission sur un lien. Cette description doit pouvoir être acquise rapidement avec un taux de fiabilité suffisant. Une approche simple est nécessaire afin de pouvoir être facilement implémentée avec les capacités réduites des capteurs, et surtout en s'appuyant sur peu de transmissions. Cela nous conduira à la deuxième partie de cette thèse.

Première partie

Routage efficace et garanti

Chapitre 2

LRP, *The Lightweight Routing Protocol*

Sommaire

Arrivé récemment dans un domaine déjà bien exploré, LRP (*Lightweight Routing Protocol*) est inspiré de deux protocoles, RPL pour son arbre de collecte et AODV (ou LOADng) pour sa recherche réactive des routes d'hôtes. En témoigne p. ex. le nom des messages utilisés. Toutefois, il n'en garde que l'idée générale, et ne leur ressemble plus beaucoup.

Les premières idées de ce protocole ont été présentées par LA, HEUSSE et DUDA [97], et se présentaient alors comme une extension de RPL. Le principe de dualité proactif-réactif, était déjà présent, ainsi qu'une ébauche de mécanisme de réparation locale (appelé alors renversement des liens, de l'anglais *link reversal*). Le protocole a ensuite été présenté pour la première fois sous ce nom par VARGA et al. [108], ils étudiaient alors IEEE 802.15.4 dans son mode de transmission avec balises (*beacon-enabled*) qui utilise déjà une structure d'arbre au niveau de la couche liaison de données. Enfin, il a été étendu et détaillé dans l'un de mes papiers publié au cours de ma thèse [1].

Présentation & motivations

Une question posée en 2015 lors de ma soutenance de master au sujet de protocole, tenait à peu près en ces mots : « *qu'est-ce qui permet de penser que LRP pourra survivre face aux innombrables protocoles de routage pour réseaux de capteurs ?* »[1]. La réponse d'alors est la même que celle d'aujourd'hui : son fonctionnement silencieux, avec une quantité de messages de contrôle limitée.

LRP propose un service de routage, tout en minimisant la surcharge due à l'algorithme lui-même — d'où le nom du protocole. Il n'est ni seulement réactif ni seulement proactif, il est en fait les deux à la fois. L'objectif est de trouver un équilibre entre ces deux approches pour se comporter de la meilleure façon possible en fonction des exigences de la situation courante. Lors de l'établissement initial de route, tant routes par défaut que routes d'hôtes, l'approche proactive est privilégiée. L'intérêt à ce moment-là est d'avoir une structure de réseau établie et fonctionnelle, pour pouvoir offrir un service de routage dès que possible. Cette structure sert également pour augmenter l'efficacité des mécanismes de maintenance des routes. Une fois cette structure établie, l'approche réactive est privilégiée pour sa maintenance. De cette façon, le protocole arrive à être extrêmement silencieux tant qu'aucune modification importante de la topologie ne survient[2]. Si un tel changement arrive, l'algorithme est capable de réagir et de trouver des solutions pour continuer à remplir son service, au prix d'un léger surcoût dû à la réparation.

LRP fonctionne à l'aide d'un arbre de collecte (cf. section 1.3.3). Il se décompose principalement en 3 grands mécanismes.

— La **collecte du trafic** de type *multipoint-to-point*. Ce type de trafic est routé de la même façon quelque soit la source du paquet, et utilise une seule route appelée **route par défaut**, l'ensemble des routes par défaut installés dans les tables de routage des différents nœuds formant un arbre enraciné sur le puits. La section 2.1.1 décrit ce mécanisme, et l'on y étudie précisément sa construction et l'association des nœuds, deux parties proactives (sections 2.1.1.1 & 2.1.1.2), puis sa maintenance, déclenchée réactivement (sections 2.1.1.3 & 2.1.1.4).

— La **distribution du trafic** de type *point-to-multipoint*. L'établissement des **routes d'hôtes** nécessaires pour ce type de trafic s'appuie sur la structure de l'arbre de collecte déjà présent. Ce mécanisme sera décrit en section 2.1.2, où l'on y étudie précisément l'établissement initial proactif des routes d'hôtes (section 2.1.2.2), ainsi que le principe de la recherche réactive

1. Jean-Luc Richier, juin 2015

2. En fait même complètement silencieux, si l'on ne regarde que le trafic de routage — mais d'autres types de trafic régulier pourraient quand même exister, comme p. ex. le trafic lié au NUD (*neighbor unreachability detection*), ou le trafic de synchronisation de la couche MAC.

d'un hôte dans le réseau (section 2.1.2.1).

— **La validation de l'acheminement**, servant de protection contre les boucles de routage. Au lieu de garantir leur absence totale à chaque instant, LRP détecte la boucle au moment où elle devrait être utilisée par un paquet, et la supprime à ce moment-là. La section 2.1.3 décrit ce mécanisme en détaillant la détection (section 2.1.3.1) et la suppression des boucles (section 2.1.3.2).

LRP supporte aussi sans plus de travail le trafic de type « *point-to-point* », où les paquets sont routés entre deux nœuds quelconques du réseau, en utilisant les routes par défaut déjà établies et ensuite les routes d'hôtes lorsque celles-ci sont atteintes, au niveau du puits ou avant.

Travaux sur LRP durant cette thèse

Le travail présenté ici au sujet du protocole LRP s'appuie en grande partie sur des travaux que j'ai déjà présenté dans des colloques nationaux et internationaux durant ma thèse — forcément résumé là et plus détaillé ici. Lors de la conférence **WiMob** (*International Conference on Wireless and Mobile Computing, Networking and Communications*), en 2015 à Abu Dhabi [1], j'ai présenté les grandes lignes directrices de LRP. Cet article explicite aussi le principe de la réparation locale (cf. section 2.1.1.4), insistant sur le fait que l'arbre de collecte est maintenu sans boucle à tout instant. Enfin, j'ai présenté le principe de validation de l'acheminement (cf. section 2.1.3), et j'ai donné la démonstration analytique de son fonctionnement. Cet article est accompagné de simulations et d'expériences mettant en œuvre et montrant les performances obtenues par les mécanismes présentés dans le papier. Elles seront reprises dans le chapitre 3. Une comparaison est faite aux résultats qu'obtient RPL dans les mêmes conditions.

Ensuite, j'ai présenté quelques améliorations significatives lors de la conférence française **ALGOTEL** (Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications) en 2016 à Bayonne [2]. J'ai présenté l'utilisation de la ERS (*Expanding Ring Search*) lors de la réparation locale (cf. section 2.1.1.4), ainsi qu'une méthode d'exploration de l'environnement d'un nœud (cf. section 2.1.1.2).

Enfin, j'ai présenté un troisième papier à la conférence **ITC** (*International Teletraffic Congress*) en 2017 à Gênes [3]. Il reprend tout d'abord les deux propositions présentées à ALGOTEL en insistant sur son silence, l'une des lignes directrices de LRP. Un troisième mécanisme leur est adjoint, celui de la réparation proactive d'une route d'hôte (section 2.1.2.2), dont l'objectif est de réparer les routes d'hôtes juste après avoir réparé l'arbre de collecte lors de la réparation locale. Enfin, cet article étudie comparativement le comportement de LRP et RPL à des situations où les liens sont asymétriques, et propose une amélioration significative à LRP pour gérer ce cas-là. Cette partie du travail est décrite au niveau du protocole dans la section 2.1.1.5. Des expérimentations sur la plateforme IoT-LAB assaisonnent également ce travail d'une saveur pratique ; elles seront reprises dans le chapitre 3.

La question de l'asymétrie des liens abordé dans le papier à ITC (chronologiquement le dernier des trois papiers présentés sur ce protocole) m'amène à me rapprocher de plus en plus de la couche physique, et conduira à la deuxième partie de mon travail, décrite dans le chapitre 4, où je présente une façon efficace d'estimer la qualité des liens radio.

2.1 Mécanismes du protocole

2.1.1 Collecte du trafic

Pour extraire facilement les données générées par les nœuds hors du réseau de capteurs, LRP organise les nœuds sous forme d'arbre de collecte (cf. section 1.3.3). Cet arbre de collecte est maintenu sans boucle à tout instant, ce qui sert non seulement directement pour un routage correct jusqu'au puits, mais aussi pour garantir la détection des boucles entre les routes d'hôtes et les routes par défaut (cf. section 2.1.3).

2.1.1.1 Construction de l'arbre de collecte

LRP est un protocole de routage à vecteur de distance (cf. section 1.3.2) qui utilise l'algorithme de Bellman-Ford distribué [52, 53] pour construire l'arbre de collecte. Un nœud annonce sa position dans l'arbre à ses voisins en diffusant un message DIO (*DODAG Information Object*), qui contient la position de ce nœud dans l'arbre. La position est décrite par trois champs. Tout d'abord, l'**identifiant de l'arbre de collecte**, permettant d'éviter des interactions gênantes entre deux réseaux cohabitant (en pratique, il s'agit de l'adresse du puits du réseau, racine de l'arbre de collecte). Ensuite, le **numéro de séquence de l'arbre de collecte**, qui peut être incrémenté par le puits afin de dater l'information transportée par le DIO, quelque peu à la façon d'un numéro de version. Il est utilisé lors des réparations globales (cf. section 2.1.1.3). Son incrément et sa comparaison suivent quelques règles spéciales décrites dans la section 2.3.4. Ces deux champs sont fixés par le puits lors de l'initialisation de l'arbre de collecte, et ne sont modifiés par aucun nœud du réseau : chaque nœud utilise l'identifiant et le numéro de séquence reçu de son successeur, c'est-à-dire du prochain saut (*next-hop*) qu'il a choisi sur sa route par défaut. Enfin, la position d'un nœud est également décrite par le **coût du chemin** qu'il utilise **pour atteindre le puits**, émetteur du premier DIO [3]. Ces trois champs forment la métrique utilisée dans LRP.

Pour comparer deux positions entre elles, les trois champs composant la métrique sont comparés dans cet ordre. L'identifiant de l'arbre, s'il diffère, implique un puits différent, donc un réseau différent, et comparer le numéro de séquence n'a alors plus aucun sens [4]. Ensuite, le numéro de séquence et le coût du chemin jusqu'au puits sont comparés d'après ce que le protocole DSDV [58] avait à l'époque proposé : un numéro de séquence plus grand implique une information plus récente, qui rend obsolète l'(ancienne) information contenue par le coût — il a donc la priorité sur le coût. Le coût sert, lorsque tous les champs précédents sont identiques, à distinguer lequel des deux chemins est le plus intéressant pour joindre le puits. Nous définissons donc une relation d'ordre entre deux positions P_i et P_j, qui est totale lorsque l'identifiant est identique : P_i est dite plus proche du puits en terme de métrique (symbole $\prec$) que P_j si et seulement si le numéro de séquence (noté `seq`) de P_i est plus grand que celui de P_j (proximité temporelle), et, en cas d'égalité, si le coût de P_i (notée `c`) est plus petit que celui de P_j (proximité spatiale, le chemin de P_i jusqu'au puits est moins coûteux que celui de P_j). Formellement,

3. La terminologie de *coût* correspond à la dépense faite par le système pour faire traverser à un paquet un lien ou les liens successifs composant un chemin. Le terme *métrique* est plus complet, combinant ici le coût du chemin jusqu'au puits aux autres informations apportées par l'identifiant et le numéro de séquence de l'arbre de collecte. La notion d'éloignement correspond ici à une position dans l'arbre par rapport au puits, toujours en termes de métrique.

4. On pourrait vouloir comparer uniquement le coût des chemins vers deux puits différents, pour choisir celui qui est le moins coûteux à atteindre ; mais cela pose d'autres contraintes (droits d'appartenance à un autre réseau, appartenance simultanée à deux arbres, gestion du changement de réseau par les sous-nœuds…) qui n'ont jusqu'ici pas été étudiées.

$$P_i \prec P_j \iff P_i.\mathtt{seq} < P_j.\mathtt{seq} \vee \big(P_i.\mathtt{seq} = P_j.\mathtt{seq} \wedge P_i.\mathtt{c} > P_j.\mathtt{c}\big) \tag{2.1}$$

En utilisant cet ordre, il est possible d'utiliser l'algorithme de Bellman-Ford distribué pour calculer, de proche en proche, la meilleure position de chaque nœud dans l'arbre. Lors de la réception d'un DIO, un nœud compare sa position courante avec celle qu'il aurait en sélectionnant l'émetteur du message comme successeur. Si la route proposée lui permet de s'approcher du puits, il a trouvé une meilleure position dans l'arbre. Il sélectionne donc ce voisin comme son nouveau successeur, (délaisse l'ancien s'il n'est pas sur un DODAG) et publie sa nouvelle position autour de lui.

L'algorithme de Bellman-Ford distribué a cependant la fâcheuse tendance à créer des boucles de routage dans les cas où un nœud s'éloigne du puits (phénomène de comptage à l'infini, cf. section 1.3.2). LRP impose donc une contrainte forte, comme quoi un nœud **ne doit en aucun cas s'éloigner du puits**. La position qu'il choisit doit être plus proche du puits que toutes les autres positions qu'il a choisies auparavant. Nous en tirons un invariant dans l'arbre : **un nœud utilise comme successeur un voisin plus proche du puits que lui-même**. Lorsque qu'un nœud i choisit j comme successeur, il se positionne lui-même plus loin du puits que j, et j ne s'éloignera jamais du puits plus loin que ce qu'il a annoncé à i. On a donc toujours $P_i \prec P_j$. Nous étudierons dans la section 2.1.1.4 les cas où la modification dans la topologie du réseau nécessite une réparation de l'arbre, et dans la section 2.1.3 les avantages qu'une contrainte aussi stricte peut nous apporter pour garantir l'absence de boucles de routage.

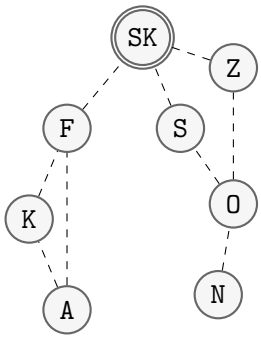
La figure 2.1 donne un exemple d'exécution pas à pas de l'algorithme de Bellman-Ford distribué. Le coût utilisé ici pour décrire un lien est hop count (cf. section 1.4.1) : on compte 1 par lien traversé. On y voit les messages DIO émis successivement par tous les nœuds de l'arbre, au fur et à mesure qu'ils se connectent. On y note que le nœud `A` n'accepte pas le DIO en provenance de `K` reçu lors de l'étape (d), parce qu'il possède déjà une position dans l'arbre plus proche du puits, en se choisissant `F` comme successeur.

2.1.1.2 Exploration de l'environnement par un nœud

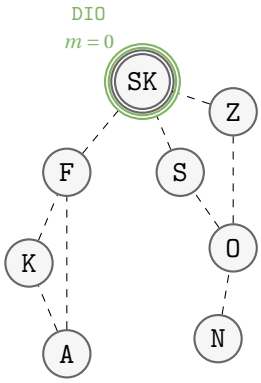
L'arbre de collecte est maintenant construit, mais LRP, à cause de son esprit réactif pour les tâches de maintenance, ne définit pas de moment régulier d'émission des DIO. Au cours de la vie du réseau, un nœud doit pouvoir explorer son environnement afin de détecter de potentiels changements dans la topologie du réseau. Par exemple, dans la figure 2.2, le lien entre le nœud `N` et le nœud `S` était absent lors de la construction de l'arbre. Pour ce faire, `N` peut diffuser un message DIO contenant sa position courante dans l'arbre par rapport au puits. Cette façon de faire a deux avantages. Tout d'abord, comme cette information est envoyée à tous les nœuds dans l'entourage direct de `N`, elle peut être prise en compte par des nœuds qui n'auraient pas reçu précédemment de message de `N` depuis la dernière mise à jour de sa position. Mais aussi, un autre nœud `S` peut détecter que `N` pourrait avoir une meilleure position dans l'arbre en le sélectionnant lui-même, `S`, comme successeur. Auquel cas, `S` peut répondre à `N` en émettant à son tour un DIO monodiffusé donnant sa propre position, pour le faire avancer dans l'arbre. C'est ce qu'on appelle un *DIO sollicité*. Les DIO dans LRP servent ainsi à la fois à annoncer la position d'un nœud, et à chacun des nœuds à sonder leur environnement [5].

On reconnait dans ce mécanisme une ressemblance avec les DIS (*DODAG Information Solliciation*) de RPL, avec quelques différences. Les nœuds RPL répondent en monodiffusion (*unicast*)

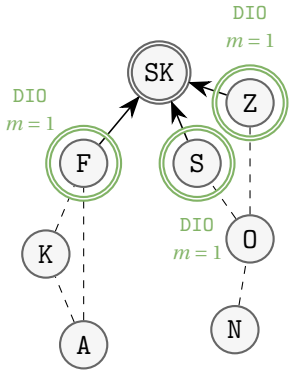
5. Dans RPL p. ex., ces deux opérations sont faites via deux messages différents, donc deux émissions différentes.



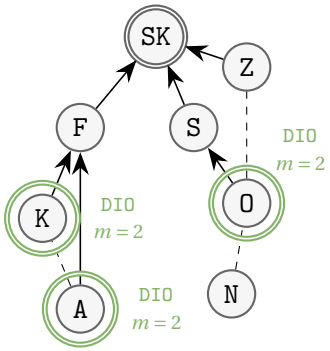
(a) Topologie initiale



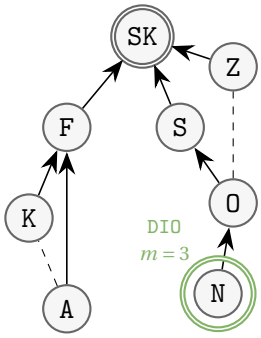
(b) Le puits (SK) initie la construction de l'arbre



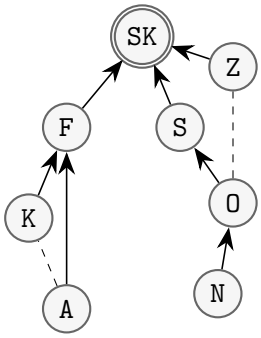
(c) Le DIO se propage dans le réseau, construisant l'arbre au fur et à mesure



(d)

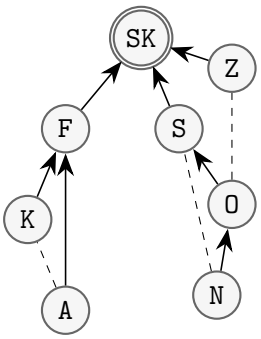


(e)

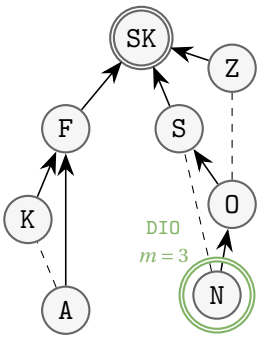


(f) arbre de collecte final

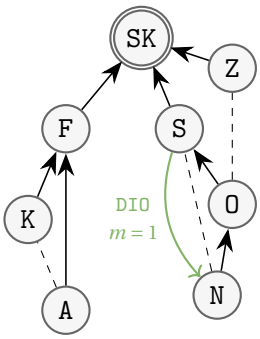
FIGURE 2.1 – Déroulement pas à pas de l'algorithme de Bellman-Ford distribué pour la construction de l'arbre de collecte. SK est le puits du réseau. Les messages DIO sont multidiffusés, ce que représente les cercles autour des nœuds émetteurs. Pour raison de clarté, seule la métrique est indiquée dans les DIO, l'identifiant et le numéro de séquence sont supposés identiques.



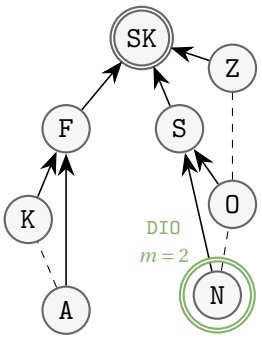
(a) Le lien N–S apparaît dans la topologie



(b) N explore son environnement

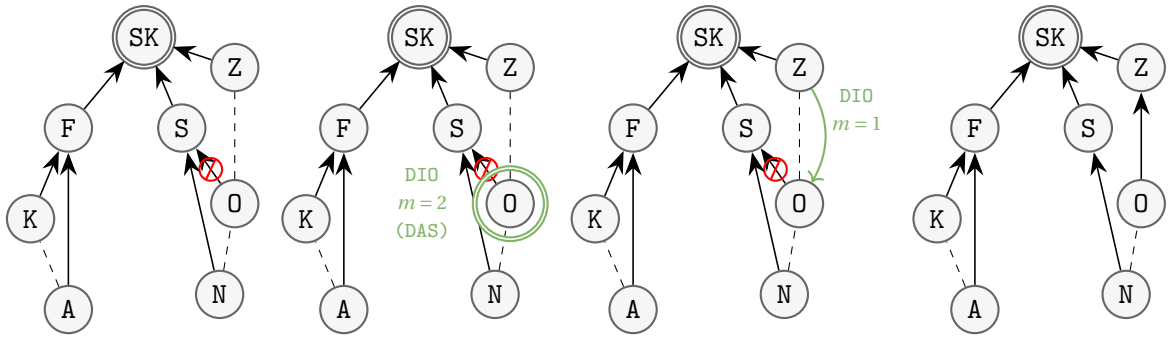


(c) S répond à travers le nouveau lien



(d) N change de position et annonce sa nouvelle position

FIGURE 2.2 – Déroulement de la découverte d'un nouveau lien. Le lien N–S n'était pas présent lors de la construction de l'arbre, dans la figure 2.1. Son apparition reflète une modification de la topologie du réseau. Lorsque N sonde son voisinage en (b), S détecte la présence de ce lien et la sous-optimalité du réseau (N peut être à une distance de 2 du puits). Il lui répond en (c) en lui envoyant un DIO sollicité.



(a) Le lien O–S est perdu

(b) O explore son environnement en sollicitant tous ses successeurs potentiels

(c) Z répond

(d) Situation finale

FIGURE 2.3 – Recherche d'un successeur alternatif dans l'arbre de collecte. Suite à une modification de topologie, le lien O–S est perdu, et O ne peut plus communiquer avec S. Il cherche un autre chemin pour atteindre le puits. Il utilise l'option DETECT_ALL_SUCCESSORS (notée DAS) pour que Z lui réponde.

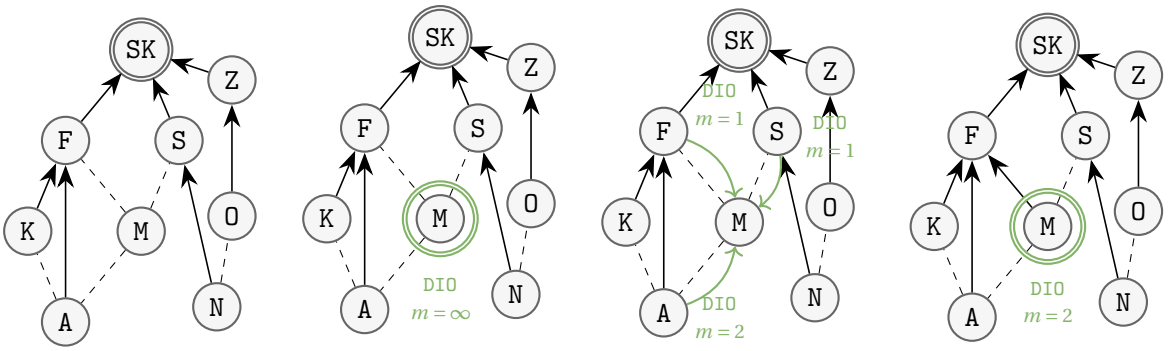
à un DIS monodiffusé, mais un DIS multidiffusé réinitialise le compteur principal de l'algorithme *Trickle*, ce qui augmente de beaucoup l'émission des DIO multidiffusés par les nœuds dans les renvoyé en monodiffusion, ce qui présente une économie de ressources (énergie et bande passante, cf. section 1.2.2). De plus, en l'état décrit par la RFC de RPL [32], il n'y a aucun moyen de préfiltrer les DIO émis par les voisins en fonction de la position qu'ils ont dans l'arbre, quoiqu'un *draft* IETF envisage cette possibilité en jouant sur le champ *Sollicited Information* contenue dans le DIS [48]. À l'inverse, dans LRP, les nœuds connaissent la position dans l'arbre de l'émetteur du DIO et peuvent eux-mêmes déterminer si leur annonce peut être intéressante. Un nœud n'enverra donc aucun DIO sollicité lorsqu'il ne peut de toute façon pas proposer un meilleur chemin que celui que l'émetteur du DIO annonce avoir — par exemple ses propres prédécesseurs, c'est-à-dire l'ensemble des nœuds qui l'ont choisi comme prochain saut de leur route par défaut.

Cette économie peut cependant être problématique dans certains cas. Notons bien la différence entre les situations suivantes : dans la figure 2.2, on ne désirait pas que O réponde à N, puisqu'il ne peut pas proposer à N une route *strictement* meilleure que celle qu'il a déjà ; mais dans la figure 2.3, une modification de topologie empêche O de communiquer avec son successeur S, et O recherche un successeur alternatif. On désire alors que Z réponde par un DIO sollicité au DIO de O. Dans ce dernier cas, il est prévu d'embarquer une option dans le DIO de O, nommée DETECT_ALL_SUCCESSORS, pour explicitement demander que tous les successeurs potentiels transmettent leur position dans l'arbre, même si la position de l'émetteur du DIO portant cette option n'en changerait pas.

Un nœud émet aussi un DIO pour sonder son voisinage lors de son initialisation. Sa position est alors une position plus éloignée du puits que tous les nœuds connectés à l'arbre de routage — une position « infiniment loin » du puits. C'est le cas de M dans l'exemple de la figure 2.4. Le DIO envoyé par M contient un identifiant d'arbre correspondant à l'adresse nulle, un numéro de séquence non significatif (valeur numérique 0, cf. section 2.3.4), et un coût du chemin jusqu'au puits de valeur maximale. Ainsi, tous les nœud dans le voisinage de M enverront un DIO sollicité en retour, et M pourra choisir, parmi eux, celui qui lui propose la meilleure route jusqu'au puits.

2.1.1.3 Réparation globale de l'arbre

La contrainte qu'un nœud ne peut en aucun cas s'éloigner du puits est imposée pour éviter les boucles de routage. Certaines situations particulières posent cependant problème. Lorsque, suite



(a) Le nœud M démarre dans le réseau

(b) M explore son environnement

(c) Les successeurs potentiels de M répondent

(d) M annonce sa nouvelle position

FIGURE 2.4 – Arrivée d'un nouveau nœud (M, au centre) dans un réseau existant. M sonde son voisinage pour obtenir la position de tous ses voisins, en diffusant un unique message DIO. Il en renvoie un autre en (d) pour annoncer sa propre position une fois qu'il est associé à l'arbre.

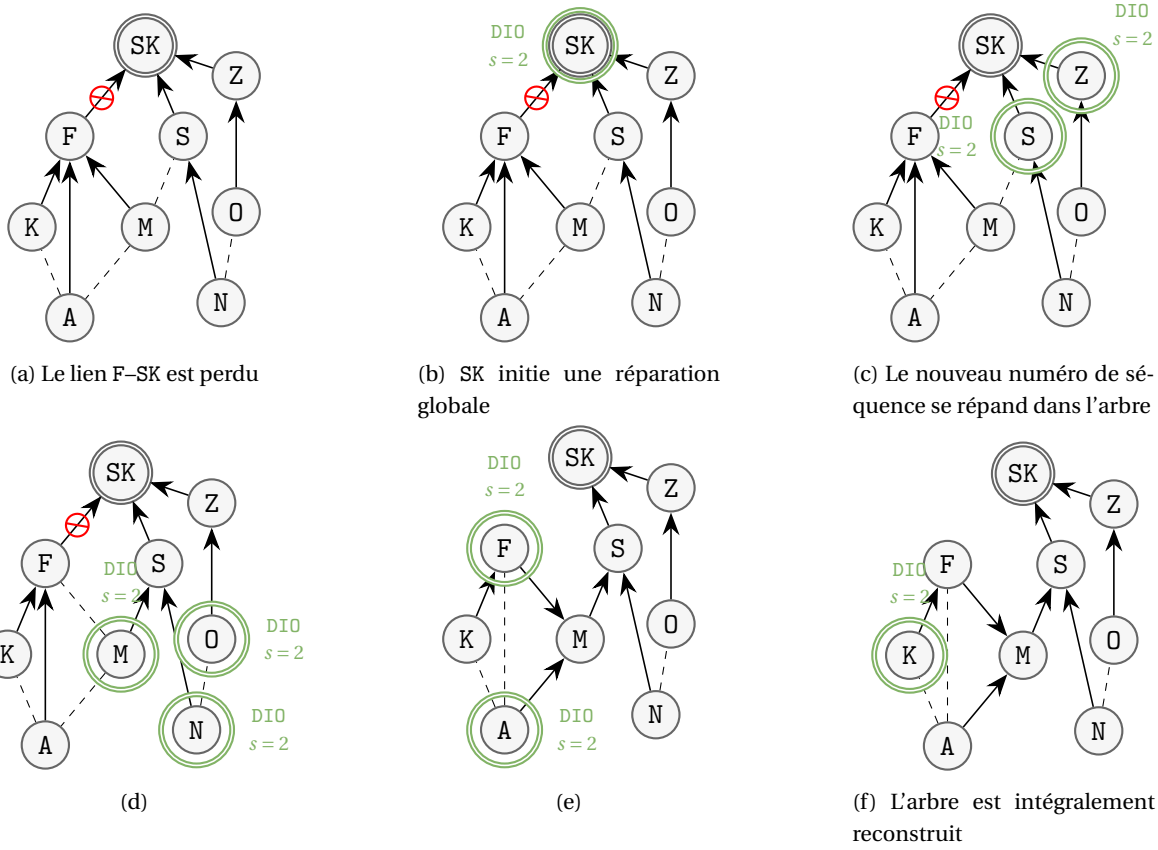
à une modification de la topologie du réseau, un nœud a perdu son successeur, il peut diffuser un DIO pour tenter de trouver, parmi ses voisins un nouveau successeur. Mais lorsqu'aucun de ses voisins n'a une position assez proche du puits pour lui proposer un chemin satisfaisant au vu de cette contrainte, il ne peut plus se raccrocher du tout au puits. C'est le cas du nœud F dans la figure 2.5. À ce moment-là, l'arbre a besoin d'être réparé. LRP propose deux mécanismes, avec leurs avantages et inconvénients respectifs : une réparation globale de l'arbre et une réparation locale.

Le mécanisme de construction de l'arbre déroulé dans les figures 2.1 et 2.5 fait naturellement suite à un incrément du numéro de séquence de cet arbre par le puits. Comme le numéro de séquence est prioritaire dans la métrique par rapport au coût du chemin jusqu'au puits, le nouveau numéro de séquence se propage dans le réseau sans que les coûts associés au nouveau numéro de séquence ne soient comparés avec ceux associés avec l'ancien numéro de séquence. L'arbre est intégralement reconstruit sans « mémoire » de l'ancien arbre : c'est une **réparation globale**. Le cas peu fréquent d'une initialisation du réseau entier[6] peut être vu comme une réparation globale, hormis le fait qu'auparavant n'existait effectivement aucun arbre de collecte.

Le déclenchement de la réparation globale correspond à la volonté de rafraîchir complètement l'arbre, pour refléter exactement la topologie courante du réseau, sans influence de la structure précédente de l'arbre. En effet, le mécanisme de réparation locale (décrite en section 2.1.1.4) peut engendrer des arbres sous-optimaux. De plus, le silence intrinsèque à LRP implique que les informations connues par les nœuds au sujet de la topologie exacte du réseau ne sont pas toujours fraîches. Reconstruire l'arbre à partir de zéro permet de résoudre ces problèmes.

Cependant, une réparation globale n'est pas une solution pouvant être utilisée de façon idéale en réponse à toutes les modifications de la topologie. En effet, son utilisation implique une diffusion de messages dans tout le réseau (au moins un message en multidiffusion par nœud), donc une augmentation temporaire de la charge du réseau et une consommation énergétique non négligeable. De plus, la réparation globale est, comme son nom l'indique, globale, c'est-à-dire qu'elle impacte l'ensemble des nœuds, alors que la plupart sont dans un état satisfaisant et qu'une petite sous-partie de l'arbre et des nœuds seulement nécessite cette réparation. Enfin, dernier point mais pas le moindre, la réparation globale ne peut être initiée que par le puits, mais la nécessité de son déclenchement n'est détectée que par les nœuds dans le réseau, qui n'ont peut-être justement

6. Il est peu fréquent dans des cas pratiques que tous les nœuds soient initialisés exactement en même temps, comme le présente une approche théorique, mais cela peut arriver, par exemple dans des plateformes de test, ou encore dans un réseau où les nœuds seraient alimentés par le même réseau électrique.



(a) Le lien F–SK est perdu

(b) SK initie une réparation globale

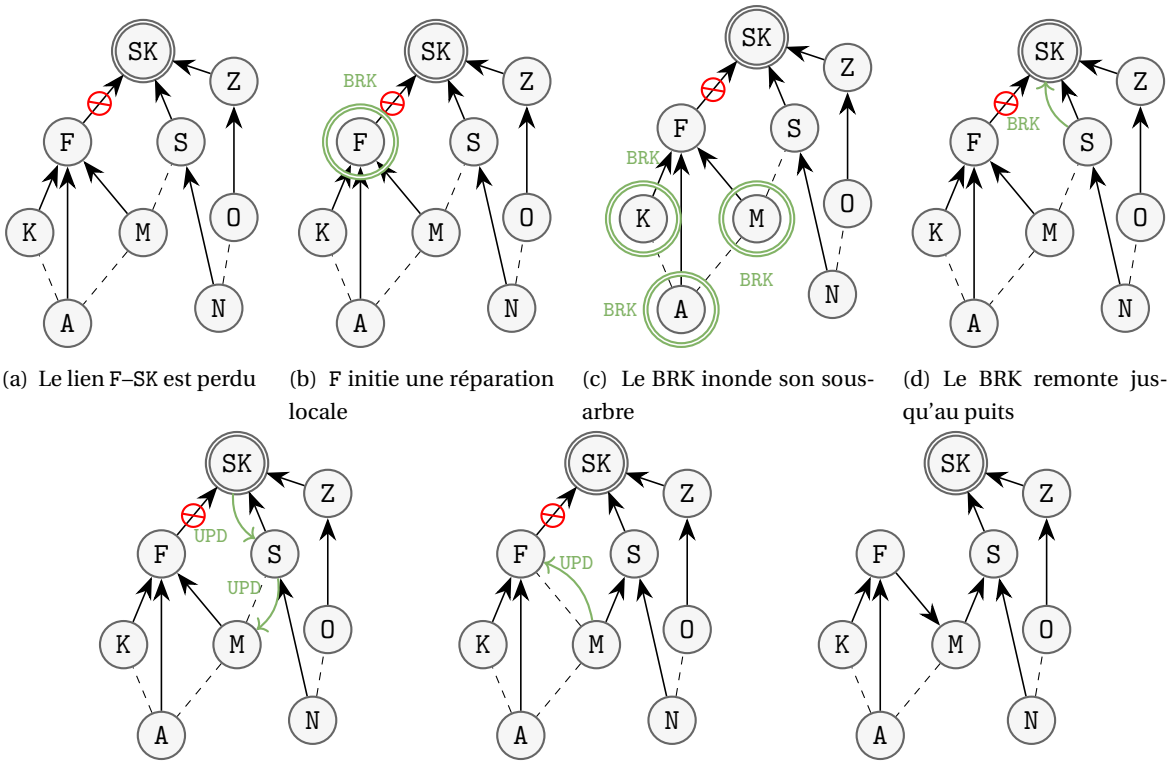
(c) Le nouveau numéro de séquence se répand dans l'arbre

(d)

(e)

(f) L'arbre est intégralement reconstruit

FIGURE 2.5 – Réparation globale de l'arbre, suite à la perte du lien F–SK. Le puits initie la réparation globale en émettant un DIO avec un numéro de séquence plus élevé que celui qui a servi à construire l'arbre précédent dans la figure 2.1. L'arbre est ensuite intégralement reconstruit. Le coût des chemins jusqu'au puits n'est pas indiqué par soucis de clarté.



(a) Le lien F–SK est perdu

(b) F initie une réparation locale

(c) Le BRK inonde son sous-arbre

(d) Le BRK remonte jusqu'au puits

(e) Le UPD suit le chemin tracé par le BRK

(f) M change de successeur

(g) F trouve un autre successeur

FIGURE 2.6 – Réparation locale de l'arbre, suite à la perte du lien F–SK. On remarque que le lien entre F et M est renversé entre l'état initial et l'état final.

plus moyen de remonter jusqu'au puits pour signaler le changement de topologie (c'est le cas de F dans la figure 2.5).

2.1.1.4 Réparation locale de l'arbre

Le mécanisme de réparation locale propose une solution complémentaire à la réparation globale. Contrairement à celle-ci et comme son nom l'indique, il n'impacte qu'un nombre restreint de nœuds en procédant localement. Il peut être déclenché par n'importe lequel des nœuds du réseau (alors appelé nœud détaché, et son sous-arbre, le sous-arbre détaché), lorsque le besoin s'en fait sentir. La figure 2.6 reprend la situation de la figure 2.5 en déroulant le mécanisme de réparation locale. Initialement, en (a), le nœud F n'a plus accès au puits, il est détaché.

Le nœud détaché tente tout d'abord de trouver un autre successeur via le mécanisme d'exploration d'environnement décrite en section 2.1.1.2 (avec l'option `DETECT_ALL_SUCCESSORS`). Si celui-ci échoue, il est contraint d'utiliser comme successeur un nœud plus éloigné du puits que lui-même. Le problème réside dans le fait que, afin de ne pas avoir de boucles de routage même temporaires, le nœud détaché doit respecter la contrainte imposée par LRP, qu'un nœud ne doit pas s'éloigner du puits. Cela signifie qu'il ne peut utiliser comme voisin qu'un nœud qu'il sait être plus proche du puits que lui-même au sens de l'équation 2.1, c'est-à-dire pour nous ici, personne. Les nœuds ont donc besoin de remonter jusqu'au puits et d'obtenir de lui un nouveau numéro de séquence capable de rafraîchir la structure courante de l'arbre. Ceci se fait en deux temps : recherche d'un chemin depuis le nœud détaché jusqu'au puits; reconstruction d'une partie de l'arbre depuis le puits sans boucle de routage.

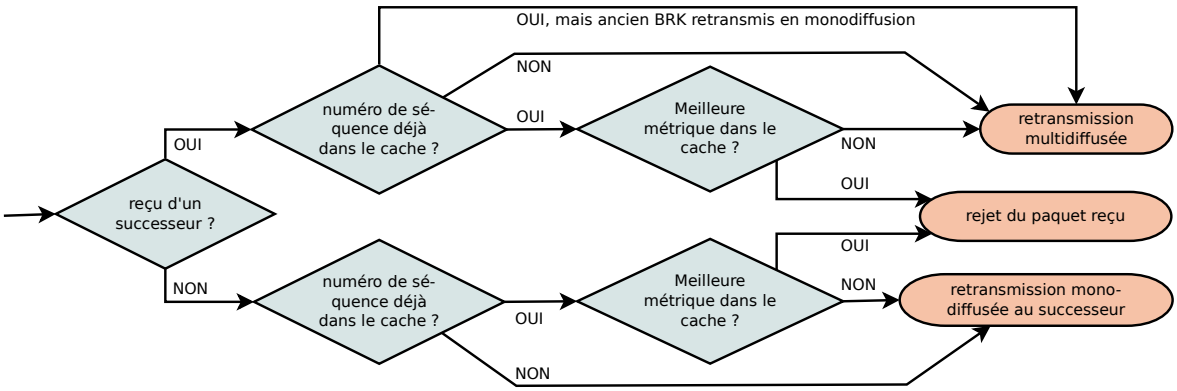


FIGURE 2.7 – Organigramme de programmation pour le traitement d'un message BRK reçu par un nœud.

Recherche d'un chemin jusqu'au puits Cette étape a deux objectifs : prévenir le puits de la nécessité d'une réparation locale, et trouver un chemin entre le puits et le nœud détaché. Pour ce faire, le nœud crée un message BRK (*Break*) et le diffuse (cf. figure 2.6b). Ce message contient la même métrique que le DIO, c'est-à-dire toutes les informations nécessaires pour décrire le chemin entre l'émetteur initial (le nœud détaché) et le nœud qui émet effectivement le message à un instant donné : l'adresse du nœud détaché et son numéro de séquence interne (fixés par le nœud détaché et modifié par aucun nœud intermédiaire) et le coût du chemin jusqu'au nœud détaché (mis à jour à chaque saut). Le BRK est transmis de proche en proche dans le réseau, en multidiffusion dans le sous-arbre du nœud, et en monodiffusion le long de l'arbre de collecte à l'extérieur. Le BRK contient donc des informations analogues à celles transportées par un DIO, mais en donnant la position de l'émetteur par rapport au nœud détaché plutôt qu'au puits.

Le schéma de la figure 2.7 donne les détails de la façon exacte dont un nœud traite ce message à sa réception. Les idées clefs en sont les suivantes. Le message est dans tous les cas comparé au cache de messages BRK que le nœud a déjà reçu. S'il a déjà reçu un BRK le positionnant plus proche du nœud détaché (au sens de l'équation 2.1), alors il existe une meilleure route pour atteindre ce nœud-là détaché : il ignore ce message-ci. Il existe une seule exception correspondant au flot le plus en haut dans la figure 2.7. Si par contre le cache contient un BRK le positionnant plus loin du nœud détaché, il lui faut alors retransmettre ce message.

Le mode de retransmission utilisé dépend de qui provient le BRK. S'il provient d'un successeur, c'est que le nœud appartient au sous-arbre du nœud détaché, et est lui aussi détaché. C'est le cas de K dans la figure 2.6. Il retransmet alors le BRK en multidiffusion (étape (b) et (c)). Si au contraire le BRK provient d'un voisin qui n'est pas un successeur, alors il y a fort à penser qu'il se trouve à l'extérieur du sous-arbre détaché. C'est le cas de S dans la même figure 2.6. Il retransmet alors le BRK en monodiffusion à son successeur.

Cette façon de procéder correspond à l'application de l'algorithme de Bellman-Ford distribué, où l'on calculerait le plus court chemin entre le nœud détaché et le puits. Dans le sous-arbre, cet algorithme s'applique directement : chaque nœud diffuse autour de lui la longueur du plus court chemin qu'il peut obtenir à travers l'un de ses voisins jusqu'au nœud détaché. À l'extérieur du sous-arbre par contre, cet algorithme est optimisé puisqu'on connaît déjà le plus court chemin pour atteindre le puits : le long de l'arbre de collecte — et c'est par là que le BRK est transmis.

Il peut arriver qu'un nœud reçoive un BRK d'un voisin qui n'est pas son successeur, mais que le nœud appartienne quand même au sous-arbre détaché. Le BRK qu'il enverra alors à son successeur sera détruit, celui-ci (ou un de ses propres successeurs, serait-ce même le nœud détaché lui-même) étant plus proche du nœud détaché. Lorsqu'il recevra par la suite un BRK avec le même numéro de séquence depuis son successeur, il devra le diffuser même si le cache contient déjà une

entrée. C'est ce cas particulier que représente le flot le plus en haut dans la figure 2.7.

Reconstruction de l'arbre depuis le puits Lorsque le puits reçoit le message BRK, il répond avec un message UPD (*Update*). Encore une fois, comme le DIO, ce message contient la position de l'émetteur du message par rapport à l'émetteur initial : l'adresse du puits (identifiant de l'arbre de collecte), un nouveau numéro de séquence de l'arbre, et le coût du chemin jusqu'au puits.

Le chemin parcouru par le BRK pour atteindre le puits est stocké par les nœuds lors de la première étape, au passage de celui-ci. Le message UPD est transmis de nœud en nœud sur ce même chemin dans le sens opposé. Le nouveau numéro de séquence du UPD est supérieur à tous ceux utilisés jusqu'ici, et supplante l'ancien numéro de séquence propagé par les DIO. Les nœuds peuvent donc changer de successeur en respectant l'invariant d'arbre : chaque nœud a comme successeur un nœud qui est plus proche du puits que lui-même (ici, en termes de numéro de séquence). En prenant l'exemple de la figure 2.6, `F`, en recevant un UPD de `M` dans l'étape (f), peut s'assurer que `M` a bien une position plus proche de `SK` que `F` lui-même n'a, quand bien même `M` était son prédécesseur au vu de l'arbre (maintenant obsolète) construit par le numéro de séquence précédent visible dans l'étape (a). L'invariant est respecté à tout instant, il n'y a jamais de boucle. On a ici un cas de reversement de lien : la route entre un successeur et son prédécesseur (entre `F` et `M`) a été renversée [7].

Le message UPD et le numéro de séquence correspondant ont cependant une portée limitée, et c'est là une différence avec le mécanisme des DIO. Le UPD n'est transmis en monodiffusion qu'aux nœuds impliqués dans la réparation locale au lieu d'être multidiffusé dans tout le réseau. C'est aussi pour cela qu'un nœud n'utilisera pas le numéro de séquence de réparation (reçu via un UPD) lors de l'émission d'un DIO — le faire aspirerait tout l'arbre vers ce nouveau numéro de séquence, inconnu du reste de l'arbre.

Il peut arriver, comme le BRK inonde l'intégralité du sous-arbre, qu'il remonte jusqu'au puits par plusieurs chemins différents. Grâce à l'adresse du nœud détaché et le numéro de séquence transportés par le BRK, les nœuds sont capables de reconnaître qu'il s'agit d'une seule et même réparation locale. Les nœuds comparent alors le coût inscrit dans le message pour connaître le meilleur des chemins, et le puits ne répondra que par un seul UPD, revenant par la route la plus courte. Il serait aussi envisageable de faire circuler plusieurs UPD sur plusieurs chemins enregistrés des BRK, afin de mitiger la sous-optimalité résultante de la réparation locale, mais cette éventualité n'a pas été creusée en profondeur dans le cadre de cette thèse.

Limitation de la profondeur de l'inondation Lorsque le nœud détaché déclenche une réparation locale, un message BRK est diffusé dans tout le sous-arbre détaché. En général, l'inondation du sous-arbre complet n'est pas nécessaire. Seulement un petit nombre de nœuds, ceux en tête du sous-arbre, sont affectés par la réparation locale. Afin de réduire le nombre de messages diffusés, LRP propose l'utilisation de la ERS, que DSR utilisait déjà en 1996 pour rechercher des routes [59], et qu'AODV utilise aussi de façon plus étendue [22]. Cette technique de recherche limite la portée de l'inondation du BRK à un nombre de sauts donné. Le nœud détaché contrôle ainsi la profondeur dans son sous-arbre jusqu'à laquelle les routes jusqu'au puits seront recherchées, et limite l'inondation à ces nœuds-là. En commençant par un anneau où lui seul est inclus (ce qui lui permet de savoir si tous ses voisins sont ses prédécesseurs ou non), il augmente petit à petit la taille de l'anneau pour atteindre des routes de plus en plus profondes. Dans la pratique, cela se traduit par l'ajout d'un champ `ring_size`, qui agit selon la même logique que le champ `time_to_live`

7. Le mécanisme de réparation locale (*Local Repair* en anglais) est parfois appelé « renversement des liens » (*Link Reversal* en anglais) — les deux appellations ont le même acronyme dans les deux langues.

d'IPv4 ou `hop_limit` d'IPv6, mais sur des paquets transmis de saut en saut en multidiffusion plutôt qu'en monodiffusion : il est décrémenté à chaque saut et, lorsqu'il atteint zéro, il inhibe la retransmission du paquet BRK.

Le choix des tailles successives des anneaux de recherche (qu'on appellera, comme quelquefois dans la littérature, ensemble de recherche) n'est pas évident. Une taille trop grande signifie que des nœuds seraient sollicités pour rien ; une taille trop petite signifie des recherches bredouilles. De plus, un tel algorithme augmente statistiquement la durée de la recherche. Plusieurs travaux ont étudié l'impact de ce choix. HASSAN et JHA [67] ont étudié le nombre de recherches L à effectuer avant de diffuser sans limite à tout le sous-arbre (c.-à-d. le rayon de l'ensemble de recherche moins 1). Ils recommandent de choisir $L \in [2,4]$ quelque soit la topologie du réseau, et annoncent une économie de 12% à 52%. DENG [71], en accord avec eux, montre qu'il existe un ensemble de recherche optimal parmi tous les ensembles de recherche de taille 3, qui est $\left\{1, \left\lfloor \frac{(H-1)^2}{2H-1} \right\rfloor, H\right\}$ (avec H le rayon du réseau) lorsque la source initiant la recherche se trouve au centre du réseau. Ces travaux donnent des indications intéressantes sur le choix de l'ensemble de recherche. Pourtant, les travaux de CHENG et HEINZELMAN [69] affirment que le gain de cette technique de recherche est négligeable lorsqu'une seule cible est recherchée à la fois, le nombre de retransmissions supplémentaires en cas de recherche trop restreinte étant statistiquement égal au nombre de retransmissions économisées par une recherche suffisamment restreinte. Dans le cas de LRP et de la réparation locale, ces résultats ne sont pas entièrement applicables. L'information recherchée par la réparation locale n'est pas aléatoirement placé dans le sous-arbre, comme le suppose les modèles de ces travaux. Au contraire, elle implique surtout la partie haute de l'arbre, proche du nœud détaché lui-même, car c'est là où se trouvent les plus courts chemins entre le puits et le nœud détaché. Dans ce cas spécifique, cet algorithme reste intéressant.

La limitation de la portée du BRK ne s'applique qu'à l'intérieur du sous-arbre. Lorsque le BRK est à l'extérieur du sous-arbre, la route pour atteindre le puits a été trouvée, et le message est monodiffusé. Le champ `ring_size` est alors ignoré.

Discussion sur la réparation locale Ce mécanisme de réparation locale limite le nombre de messages utilisés lors de la réparation mais ne recrée pas forcément un arbre de plus court chemin. Par exemple, dans la figure 2.6g, `A` n'a pas vu passer le UPD, et utilise donc toujours `F` comme successeur, alors que `M` serait un successeur plus proche du puits. Le principal avantage de cette méthode est le faible impact en messages échangés : une inondation limitée, restreinte à la tête du sous-arbre détaché, et quelques flots aller-retour de paquets monodiffusés. D'autres protocoles utilisent une sorte d'aller-retour monodiffusé vers la destination recherchée pour recréer la route perdue : TPGF (*Two-Phase geographic Greedy Forwarding*) utilise un paquet pour se promener à travers le sous-arbre détaché, jusqu'à ce qu'il trouve un chemin pour aller et revenir [84]. Dans Babel, la requête d'un numéro de séquence peut voyager tout du long jusqu'à la cible pour déclencher une mise à jour [28]. TORA [39], de son côté, propose une façon d'inverser localement le sens d'utilisation des liens [7] entre un nœud et ses voisins, jusqu'à avoir une nouvelle route fonctionnelle vers la destination. On peut remarquer aussi la similitude qui existe entre la méthode de transmission d'un BRK et celle d'un Smart-RREQ de LOADng [93] : tous deux possèdent une destination (implicitement le puits pour les BRK) ; tous deux sont transmis en multidiffusion quand aucune route vers cette destination n'existe (la route par défaut n'est pas utilisable quand le BRK est reçu d'un successeur) ; tous deux sont transmis en monodiffusion le long de la route lorsque celle-ci est présente ; tous deux enfin, une fois arrivés à destination, génèrent une réponse qui suit le tracé du message dans le sens inverse. L'esprit dans lequel la chose est réalisé est néanmoins différent. Dans le cas de LRP, c'est une route alternative vers le puits qui est recherchée. LOADng, lui, propose cette fonctionnalité comme intermédiaire entre sa conception de base (où le message

de recherche de route est contraint d'aller jusqu'au nœud cherché) et le fonctionnement du protocole AODV duquel il est inspiré (pour lequel un hôte intermédiaire peut répondre à la place de la destination finale d'un RREQ). Il est aussi à noter que, dans le cas où le nœud qui émet le Smart-RREQ (*Route Request*) est utilisé par les autres nœuds pour rejoindre la destination, ce mécanisme a l'effet d'empêcher totalement la réparation de la route.

2.1.1.5 Gestion des liens asymétriques

Déjà abordée dans la section 1.4.2, l'asymétrie des liens n'est pas un facteur négligeable, bien qu'elle reste un cas minoritaire. LRP propose un mécanisme pour connaître la qualité d'un lien dans les deux sens. Pour ce faire, un capteur échange avec son voisin un message HELLO monodiffusé. Ce message contient le coût du lien entre l'émetteur et le destinataire du message, au vu de l'émetteur. En recevant ce message, le destinataire répond par un message semblable, avec sa propre mesure de qualité du lien. Cet échange est réalisé à chaque fois que le coût du lien doit être mis à jour, soit lors de sa première détection soit à un autre moment afin de vérifier et de mettre à jour l'information acquise. Dans le cas où aucune réponse au message HELLO n'est reçue, un cas d'asymétrie totale est détecté (la communication n'est possible que dans un sens), et le nœud voisin peut être inscrit sur une liste noire pour quelques temps, afin de ne pas réagir aux messages qu'on reçoit de sa part.

Le message HELLO vient à l'origine du protocole de découverte du voisinage (NHDP, *Neighborhood Discovery Protocol*) définit par la RFC 6130, et utilisé par exemple dans LOADng-CTP [102]. Le protocole de découverte du voisinage « utilise un échange local de messages HELLO afin que chaque routeur puisse déterminer la présence et la connectivité de ses voisins à 1 saut et à 2 sauts symétriques » [29]. Le message ne peut pas être utilisé tel quel dans LRP. En effet, le mécanisme de découverte dans LRP est déjà en grande partie réalisé par les messages DIO multidiffusés dans le réseau, servant également à la construction de l'arbre. De plus, même si les HELLO tels que définis par la RFC 6130 transportent un indicateur de la qualité du lien, il est également dit explicitement que « la qualité du lien est un mécanisme d'‹ admission de lien ›, permettant à un routeur de déterminer qu'un lien donné est trop instable pour même envisager son utilisation. Il ne s'agit pas spécifiquement d'une mesure de métrique (ou de coût) de lien ni d'un substitut à une telle mesure. [...] L'information sur la qualité des liens n'est donc pas équivalente à une métrique de lien. » [29] Or, dans LRP, les messages transportent directement la qualité des liens.

Une fois la qualité du lien connue dans les deux sens, la décision de routage subséquente peut être faite en connaissance de cause et d'asymétrie potentielle. Une solution simple pour la prendre en compte est de considérer que le coût du lien est le coût le plus élevé des deux sens de propagation. De cette façon-là, on est sûr d'avoir quelque chose d'au moins aussi bon que prévu. Pour étendre la réflexion plus loin, il serait intéressant d'étudier les possibilités de routage sur des liens asymétriques, où le schéma des routes n'est pas identique dans un sens et dans l'autre. Cette possibilité peut toutefois poser des problèmes complexes, comme pour la question des acquittements qui doivent nécessairement transiter dans l'autre sens. L'étude n'a pas été poussée plus loin dans le cadre de cette thèse.

En parlant du coût exact et de l'asymétrie d'un lien, nous abordons la problématique de gestion des voisins. Il s'agit de connaître ses voisins, de détecter leur apparition et leur disparition, et de mesurer précisément les paramètres des communications radio avec chacun d'entre eux afin de déterminer des coûts décrivant leur qualité. Le discours se rapproche ainsi très fortement des couches inférieures, la couche liaison de données et la couche physique. Nous avons déjà eu l'occasion de discuter de quelques techniques existantes d'estimation des coûts des liens dans la section 1.4.1, et nous reviendrons dessus avec plus de détails dans le chapitre 4.

2.1.2 Distribution du trafic

Jusqu'ici, nous avons vu comment LRP s'organisait pour collecter le trafic montant (trafic « *multipoint-to-point* », cf. figure 1.9a). La question de la distribution du trafic répond à la même question pour le trafic descendant (trafic « *point-to-multipoint* », cf. figure 1.9b), où les paquets vont du routeur de bordure (c'est-à-dire du puits) vers n'importe quel nœud à l'intérieur du réseau. Contrairement au trafic montant où une seule route par défaut suffit, le trafic descendant a besoin d'une route d'hôte pour chaque nœud, établie entre le puits et le nœud. La section 2.1.2.1 montre comment LRP peut établir des routes d'hôtes de façon réactive, à la demande, lorsque il en est besoin, en utilisant la même approche que AODV. La section 2.1.2.2 montre que LRP peut également construire les routes d'hôtes de façon proactive, comme dans RPL, pour diminuer le nombre de messages de contrôle dans certaines situations. Enfin, la section 2.1.2.3 explique la restauration proactive des routes d'hôtes d'un sous-arbre suite à une réparation locale.

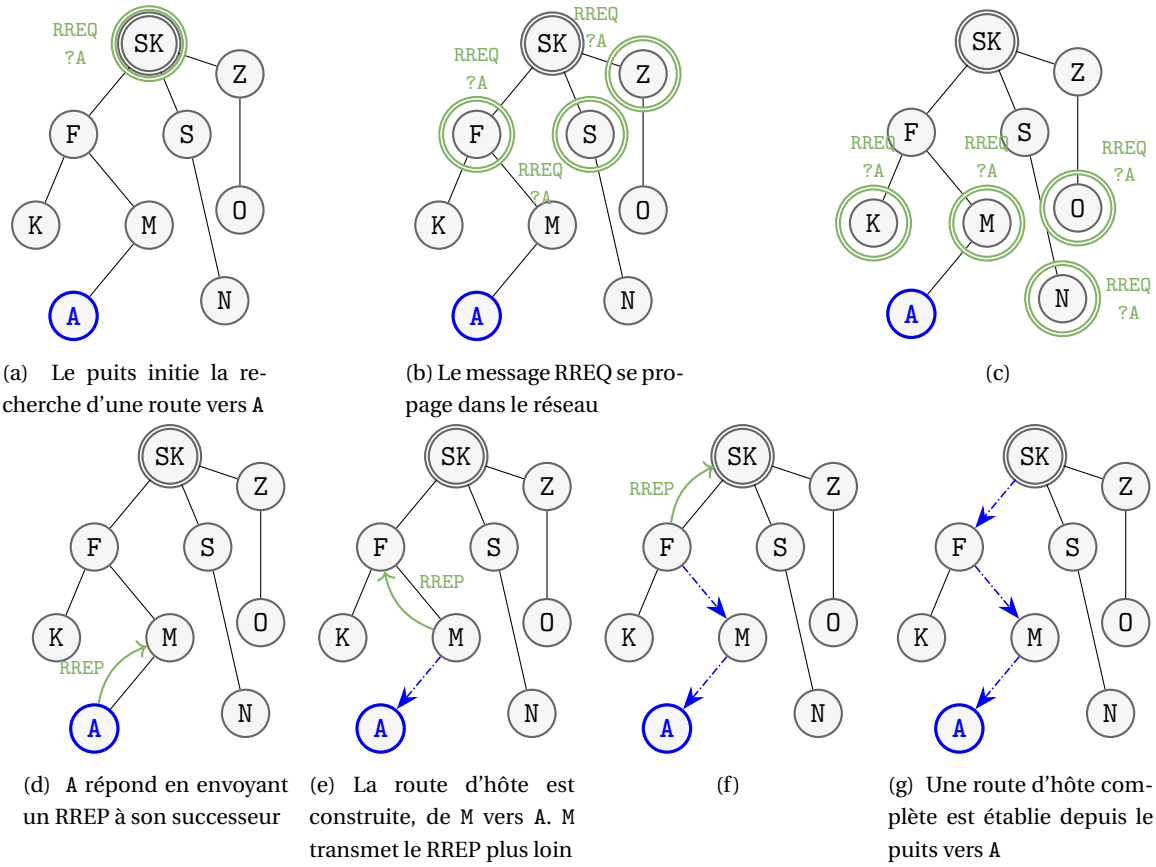
2.1.2.1 Création réactive d'une route d'hôte

Le puits utilise une route d'hôte lorsqu'il doit transmettre un paquet à un nœud à l'intérieur du réseau. Il peut arriver cependant qu'il ne possède aucune route vers ce nœud-là, parce que la route a été perdue, détruite, ou n'a pas été proactivement construite par le nœud (pour le moins pas jusqu'au puits). Dans ce cas, le puits doit rechercher cet hôte dans le réseau ; or il ne peut justement pas utiliser la route d'hôte qui lui manque pour atteindre l'hôte et lui signaler qu'il lui manque une route pour l'atteindre ! Il inonde donc le réseau à sa recherche, en diffusant autour de lui un message RREQ, propagé dans le réseau de proche en proche. Un numéro de séquence, déterminé par le puits et embarqué dans le message, est utilisé pour identifier ce message. Le RREQ contient également l'adresse de l'hôte recherché. La figure 2.8a–c illustre cette situation.

Lorsque l'hôte recherché lui-même reçoit le message RREQ, il répond par un message RREP (*Route Response*). À l'instar d'un DIO, d'un BRK et d'un UPD le RREP possède des informations décrivant le chemin jusqu'à l'émetteur initial du RREP : son adresse (`A` dans l'exemple de la figure 2.8), un numéro de séquence fixé par celui-ci, et le coût du chemin jusqu'à lui. Il envoie le RREP à son successeur. Celui-ci crée dans sa table de routage une route à destination de cet hôte-là, dont le prochain saut est le voisin duquel il a reçu le RREP, et le transmet à son propre successeur. Et ainsi de suite, jusqu'à ce que le message atteigne le puits. La route d'hôte est alors créée de bout en bout, du puits jusqu'à l'hôte.

La transmission d'un RREP est soumise à certaines vérifications, afin de garantir l'état du réseau. Tout d'abord, de la même manière que lors de la sélection des successeurs dans l'arbre, il est interdit aux nœuds de s'éloigner d'un hôte, d'après la relation d'ordre définie dans l'équation 2.1. Le même invariant sert alors à garantir l'absence de boucles dans la route d'hôte : le prochain saut d'un nœud sur la route d'un hôte est toujours plus proche de l'hôte que le nœud lui-même. Un nœud connait la position de son voisin sur cette route lors de la réception d'un RREP, et vérifie la validité de l'invariant avant d'accepter le RREP — un numéro de séquence plus ancien sera par exemple refusé. Une autre contrainte est qu'un nœud ne doit jamais envoyer un message RREP à un voisin s'il n'est pas sûr que ce voisin-là est plus proche dans l'arbre du puits que lui-même. Cette condition est garantie par construction de l'arbre si le voisin est le successeur du nœud — et en pratique, c'est seulement à lui qu'un nœud envoie des RREP. Cette deuxième contrainte est nécessaire pour garantir le fonctionnement de la validation de l'acheminement (cf. section 2.1.3).

Si ce mécanisme de recherche réactive est nécessaire pour gérer les cas où la route d'hôte est absente au niveau du puits, elle n'est pas idéale dans toutes les situations, car elle génère une grande quantité de trafic coûteux multidiffusé (cf. section 1.2). L'impact de l'inondation déclenchée pourrait être atténué en réduisant le nombre total de RREQ émis, par exemple transformant



(a) Le puits initie la recherche d'une route vers A

(b) Le message RREQ se propage dans le réseau

(c)

(d) A répond en envoyant un RREP à son successeur

(e) La route d'hôte est construite, de M vers A. M transmet le RREP plus loin

(f)

(g) Une route d'hôte complète est établie depuis le puits vers A

FIGURE 2.8 – Recherche active d'un nœud (le nœud A) dans le réseau. L'arbre de collecte sous-jacent est celui de la figure 2.4d. Le puits initie la recherche en émettant un message RREQ contenant l'adresse de A. Ce message est transmis de successeur en prédécesseur tout le long de l'arbre. A répond et construit sa route d'hôte en émettant un message RREP à destination du puits.

les multidiffusions en monodiffusions lorsqu'une route est connue pour l'hôte recherché — c'est l'objectif des *Smart Route Requests* de LOADng [93]. Mais l'idéal est même d'éliminer complètement cette inondation lorsque cela est possible, et c'est le but de la création proactive des routes d'hôtes.

2.1.2.2 Création proactive d'une route d'hôte

Lorsque le nœud change de successeur parce que l'ancien successeur est hors d'atteinte — ou à plus forte raison lorsqu'il vient de s'associer au réseau — il y a fort à penser que sa route d'hôte ne soit plus utilisable. Il sait donc que, lorsque le puits voudra lui envoyer du trafic, il devra inonder le réseau pour l'y rechercher comme décrit dans la section précédente. Afin d'éviter cette inondation, il construit proactivement sa route d'hôte en envoyant spontanément un message RREP au puits par l'intermédiaire de son nouveau successeur. De cette façon, la route d'hôte est créée sans nécessiter d'inonder le réseau. Dans le cas de l'exemple de la figure 2.8, seules les étapes (d–g) se déroulent alors.

2.1.2.3 Restauration proactive d'un ensemble de routes d'hôtes

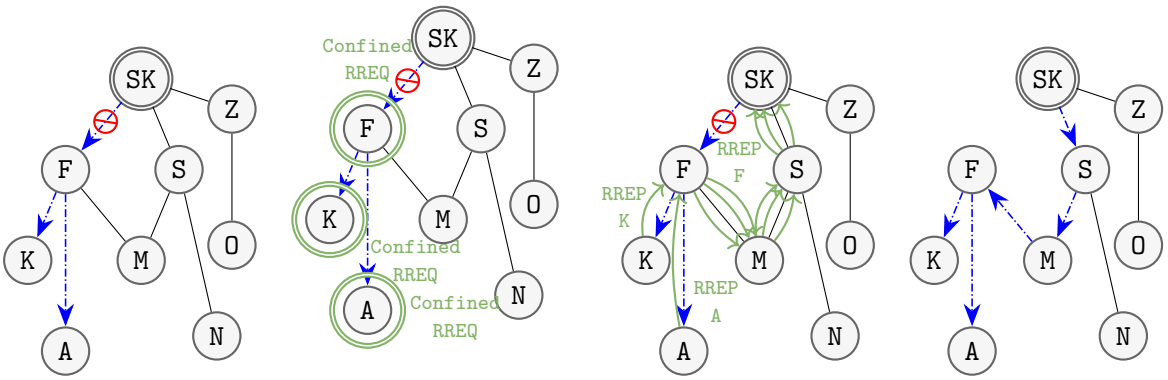
Cette façon de reconstruire proactivement la route d'hôte est valable pour un nœud qui change de successeur lui-même. Le problème qui suit est l'ensemble des nœuds de son sous-arbre, dont les routes d'hôtes ne sont également plus utilisables. C'est le cas de `F` et de son sous-arbre, `K` et `A` dans la figure 2.9a. À plus forte raison à cause de leur grand nombre, il est fort désirable d'éviter d'inonder l'intégralité de l'arbre pour reconstruire ces routes connues à l'avance comme n'étant plus utilisables. Une solution naïve serait que le nœud `F` transmette un RREP au nom des nœuds de son sous-arbre, car il connait la route pour atteindre chacun d'eux, mais, il ne peut pas modifier le numéro de séquence qu'il a reçu auparavant avec le RREP de l'un d'eux lors de l'établissement de sa route. Le seul RREP qu'il pourra générer en son nom utilisera le même numéro de séquence que celui que ses successeurs successifs et le puits utilisaient. Ce RREP décrira très certainement un chemin plus coûteux jusqu'à l'hôte, et le respect de l'invariant le rendra inacceptable.

Pour pouvoir remplacer la route, il faut donc impérativement que l'hôte génère lui-même un nouveau numéro de séquence rendant ainsi obsolète l'ancien. C'est le but du RREQ confiné (*Confined RREQ*), visible dans la figure 2.9b. Ce RREQ spécial peut être généré par n'importe quel nœud de l'arbre. Comme un RREQ standard, chaque nœud qui le reçoit le transmet plus loin ; cependant, le drapeau `CONFINED` qu'il transporte indique qu'il doit être pris en compte uniquement lorsqu'un nœud le reçoit de la part de son successeur — le message est donc confiné au sous-arbre de l'émetteur. Le message ne cible pas un nœud particulier ; au contraire, il indique que la route déservant n'importe quel nœud qui reçoit ce message doit être reconstruite proactivement (figure 2.9c–d), en utilisant un nouveau numéro de séquence.

Dans une réparation locale, plusieurs nœuds changent de successeur. Dans l'exemple de la figure 2.6, `F` et `M` ont tous deux changé de successeur. Mais on voudrait que seul le sommet du nouveau sous-arbre (`M`) diffuse un RREQ confiné, afin qu'il n'y ait qu'une seule inondation. Un nœud se reconnait tel parce qu'il a reçu le message UPD d'un voisin qui n'était ni son successeur ni son prédécesseur : il était en-dehors du sous-arbre détaché. Ce seul nœud émettra donc un RREQ confiné, et les autres nœuds attendront de le recevoir pour le transmettre à leur tour.

2.1.2.4 Trafic point à point

Le trafic le plus attendu dans un réseau de capteurs est le trafic montant. Il correspond typiquement à des données générées par les capteurs. Il est extrait du réseau le long de l'arbre de



(a) Après la réparation locale, les routes d'hôtes sont inutilisables.

(b) F signale la situation à tout son sous-arbre en diffusant en RREQ confiné.

(c) Les RREP spontanés remontent le long de l'arbre de collecte.

(d) Toutes les routes d'hôtes sont finalement reconstruites.

FIGURE 2.9 – Établissement proactif des routes d'hôtes de tout un sous-arbre suite à la réparation locale de l'arbre de collecte en figure 2.6g.

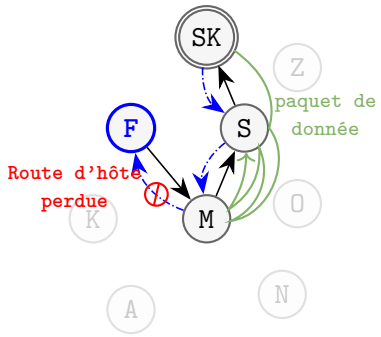
collecte. Le trafic descendant correspond plutôt à du trafic de gestion des nœuds (configuration, accusés de réception…) ou lorsque des actionneurs sont présents. Il est distribué dans le réseau grâce aux routes d'hôtes. Le trafic point à point, quant à lui, correspond à une communication entre les capteurs, afin d'échanger une information sans passer par un service extérieur au réseau. Un exemple simple est le cas d'un interrupteur connecté devant configurer l'allumage d'un plafonnier.

Ce dernier type de trafic est supporté par LRP sans autre mécanismes que ceux utilisés pour la collecte et la distribution du trafic. Les nœuds utilisent la route par défaut, et le paquet envoyé remonte ainsi l'arbre de collecte jusqu'à ce qu'il atteigne la route d'hôte de la destination, qu'il suit jusqu'au bout, à l'instar du protocole RPL. Si aucune route d'hôte n'est trouvée, le paquet remontera jusqu'au puits, qui déclenchera le mécanisme de recherche réactive de route (cf. section 2.1.2.1), à l'instar du protocole AODV.

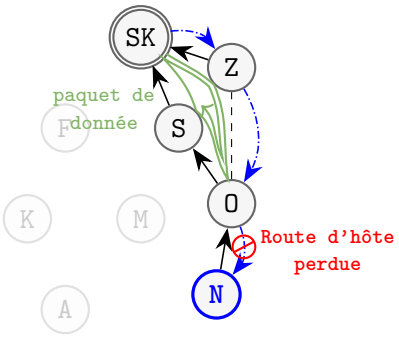
L'utilisation d'un arbre de collecte et de routes d'hôtes rend ainsi triviale la gestion de ce type de trafic. On pourrait lui reprocher sa sous-optimalité : le trafic n'est pas capable de traverser le réseau en « diagonale », en sautant d'une branche à l'autre, mais est au contraire obligé de remonter vers le cœur du réseau avant de redescendre le long d'une branche de l'arbre. Ceci représente une plus grande dépense de ressources puisque la route utilisée n'est pas la plus courte, et charge plus les nœuds au cœur du réseau. Un routage totalement maillé serait alors plus efficace. C'est l'approche qu'utilise AODV par exemple, au prix d'une augmentation du coût de l'établissement des routes entre chaque hôte dans le réseau. À l'inverse, l'arbre de collecte utilisé par LRP permet de diminuer le nombre de routes calculées en agrégeant le trafic sur une seule route, déjà calculée. De plus, ce type de trafic est peu attendu en réseau de capteurs, ce qui réduit l'impact du surcoût de transmission.

2.1.3 Validation de l'acheminement

Dans un réseau LRP, les routes par défaut et les routes d'hôtes coexistent. Lorsque les tables de routage ne sont pas cohérentes, cette coexistence peut créer des boucles de routage. La figure 2.10a montre un exemple trivial de boucle de routage. Elle a pu apparaître simplement après la destruction d'une route d'hôte, quelque part dans l'arbre. Même si le cas semble facile à détecter, parce qu'il n'implique que deux hôtes, il peut aussi arriver des cas où plusieurs hôtes sont impliqués. C'est le cas de la figure 2.10b. Une telle situation peut arriver de la même manière,



(a) La route de M vers l'hôte F est perdue. M utilise sa route par défaut, et S utilise la route d'hôte.



(b) La route de O vers l'hôte N est perdue. O et S utilisent la route par défaut, et SK et Z utilisent la route d'hôte.

FIGURE 2.10 – Exemples de boucles de routage. Dans les deux cas, le paquet de donnée tourne infiniment (jusqu'à expiration de son champ `time_to_live`) entre les nœuds sans jamais arriver à destination.

lorsque O perd sa route vers N ; mais ici, il a auparavant changé de successeur, et ne renverra donc pas le paquet au même voisin que celui de qui il a reçu le paquet. La boucle implique 4 nœuds, et est beaucoup plus difficile à détecter.

LRP ne tente pas de garder une cohérence parfaite entre les tables de routage des nœuds. Garder un système cohérent implique l'ajout de contraintes et de surcharges de communication, que l'on voudrait particulièrement éviter dans le cadre d'un réseau de capteurs sans fil. LRP a de plus une facette réactive, lui permettant de réagir lorsqu'une incohérence est détectée. Le travail consiste donc simplement à détecter, puis à supprimer les boucles de routage avant leur utilisation.

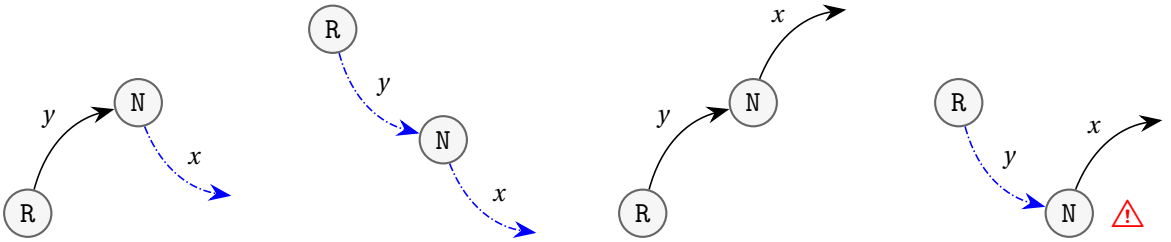
2.1.3.1 Détection d'une boucle

Lors du routage d'un paquet, un nœud utilise soit une route spécifique à l'hôte destinataire du paquet, construite par la transmission d'un RREP, soit une route par défaut le long de l'arbre de collecte, construite par la réception d'un DIO. Dans tous les cas, cette route a été construite avec un numéro de séquence (seq), un coût de chemin jusqu'à la destination (c), une longueur de préfixe (plen, qui permet de distinguer les routes d'hôtes de la route par défaut), et un prochain saut. Soit l'ordre noté $\prec$ sur toutes les entrées de table de routage x et y dans le réseau : $x \prec y$ si et seulement si x est plus *spécifique*, plus *récente*, ou a un chemin vers la destination moins coûteux que y, dans cet ordre de priorité. Formellement,

$$\begin{aligned} x \prec y \overset{\text{def}}{\iff} & x.\texttt{plen} > y.\texttt{plen} \vee \\ & x.\texttt{plen} = y.\texttt{plen} \wedge (x.\texttt{seq} > y.\texttt{seq} \vee \\ & \qquad x.\texttt{seq} = y.\texttt{seq} \wedge x.\texttt{c} < y.\texttt{c}) \end{aligned} \tag{2.2}$$

Cette relation d'ordre est totale. L'absence de boucles est garantie si les entrées dans les tables de routage utilisées le long du cheminement d'un paquet dans le réseau jusqu'à la destination sont strictement décroissantes, c'est-à-dire si on se rapproche de la destination en termes de spécificité, de récence, ou de proximité spatiale. La difficulté est de connaître, au moment du routage d'un paquet dans un nœud, la route utilisée par le nœud précédent.

Soit donc un nœud N, devant router un paquet en utilisant la route x, en provenance d'un nœud R où il a été transmis conformément à l'entrée y. On veut que $x \prec y$ quelque soit la situation. Quatre combinaisons s'offrent à nous, synthétisées dans la figure 2.11 et décrites ci-dessous.



(a) Une route par défaut suivie d'une route d'hôte

(b) Une route d'hôte suivie d'une route d'hôte

(c) Une route par défaut suivie d'une route par défaut

(d) Une route d'hôte suivie d'une route par défaut

FIGURE 2.11 – Quatre cas possibles de succession de deux routes, empruntées par un paquet. Le cas (d) pose problème, car on ne respecte pas la condition $x \prec y$.

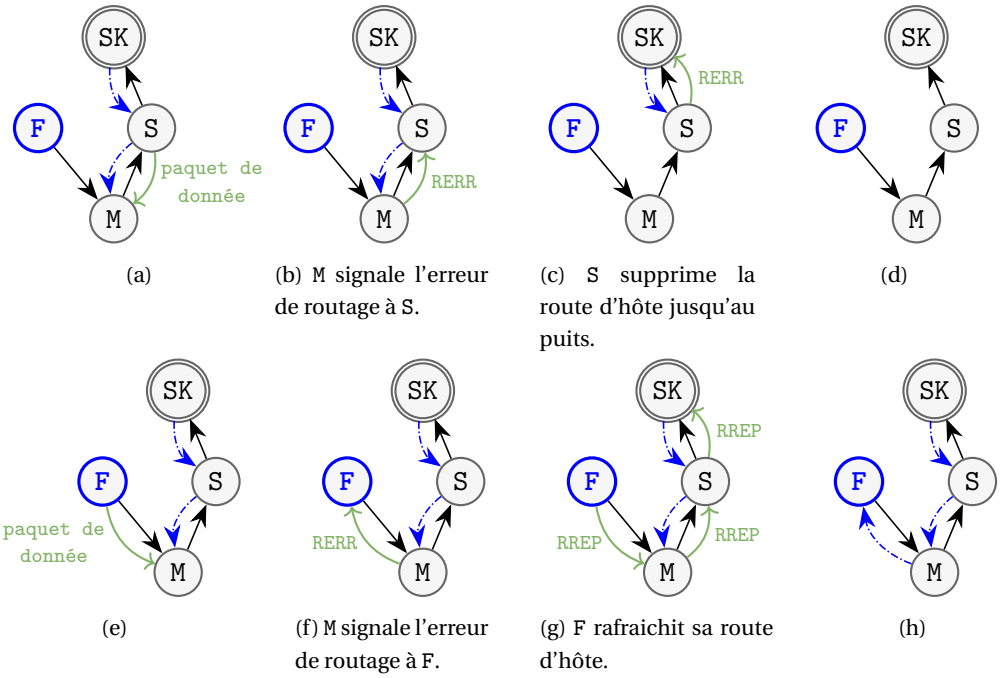
— **Lorsque x est une route d'hôte et y une route par défaut** (cf. figure 2.11a), on a $x \prec y$ directement à cause de la longueur du préfixe des routes. En pratique, dans ce cas-là, il s'agit d'un paquet point à point (cf. section 2.1.2.4), et `N` est le nœud à partir duquel, après avoir remonté dans l'arbre, le paquet commence à descendre vers la destination.

— **Lorsque x et y sont tous deux une route d'hôte**, (cf. figure 2.11b) ou **tous deux une route par défaut** (cf. figure 2.11c), on a alors x.plen = y.plen. De plus, on sait que la route y a été construite dans `R` à la réception d'un message (respectivement un DIO ou un RREP) de la part de `N`, transportant un numéro de séquence un coût du chemin jusqu'à la destination. `N` peut avoir entre temps reçu un nouveau message contenant un nouveau numéro de séquence que `R` n'a pas reçu, mais, par respect de la contrainte imposée par LRP dans les section (resp.) 2.1.1.1 et 2.1.2.1, on sait qu'il n'a pas accepté de numéro de séquence plus ancien. `R`, quant à lui, n'a pas changé de numéro de séquence, car il aurait alors aussi dû changer de prochain saut pour cette route. On a donc x.seq $\geq y$.seq. Pour le coût du chemin, on a par construction x.m = y.m + l, avec l, le coût du lien de `N` à `R`, toujours strictement positif. On a donc toujours $x \prec y$.

— **Lorsque x est une route par défaut et y une route d'hôte** (cf. figure 2.11d), on a au contraire $x \succ y$ à cause de la longueur de la route. `N` ne doit alors pas transmettre le paquet, parce qu'il est fort probable qu'il l'envoie dans une boucle.

Afin de détecter ce cas problématique, deux règles sont mises en place, l'une devant être respectée par `R` et l'autre par `N`. On remarque que les routes d'hôtes sont toujours construites dans le sens inverse de la route par défaut (les RREP ne sont envoyés qu'au successeur). La première règle dit que `R`, lorsqu'il choisit un nouveau successeur, doit **supprimer toutes les routes d'hôtes qu'il possède et dont le nouveau successeur est le prochain saut** (règle 1). Cette règle permet de garantir qu'une route d'hôte n'est pas construite dans le sens opposé d'une autre route d'hôte (de `R` vers `N` et de `N` vers `R`). L'absence de boucles dans l'arbre de collecte nous permet de garantir qu'il n'y a jamais deux routes par défaut en sens opposé, et ainsi que deux routes d'hôtes n'ont pas non plus été créées simultanément dans le sens opposé.

Une fois ceci acquis, `N` peut distinguer, lorsqu'il doit router un paquet sur sa route par défaut, entre les cas des figures 2.11c et 2.11d. Comme les routes d'hôtes ne peuvent pas exister dans les deux sens, **`N` doit s'assurer de ne pas être dans le cas 2.11d en vérifiant qu'il a une route d'hôte** (quelconque) **envoyant vers le prochain saut `R`** (règle 2). C'est naturellement le cas pour tous ses prédécesseurs.



(a) (b) M signale l'erreur de routage à S. (c) S supprime la route d'hôte jusqu'au puits. (d)

(e) (f) M signale l'erreur de routage à F. (g) F rafraichit sa route d'hôte. (h)

FIGURE 2.12 – Gestion de la route d'hôte en direction de F, obsolète à cause de sa perte entre M et F. Dans les deux cas (les deux lignes d'images), le paquet de données permettant la détection arrive de deux nœuds différents, mais M détecte l'erreur de routage de la même façon, à l'aide de la règle 2 de la détection de boucles.

2.1.3.2 Suppression d'une boucle

La détection des boucles décrite ci-dessus ne permet pas de distinguer entre un paquet qui entre effectivement dans une boucle et un prédécesseur légitime mais inconnu. La figure 2.12, reprend dans la figure 2.10a en présentant ces deux cas (les deux lignes de la figure). En raison de la disparition de la route d'hôte de M à F, il y a une boucle de routage entre S et M pour les paquets à destination de F que M détecte bien lorsqu'il reçoit un paquet en provenance de S ; mais M détecte aussi comme incorrects les paquets en provenance de F, qui pourraient eux aussi conduire potentiellement à une boucle de routage.

Dans le cas (a–d), c'est S qui envoie le paquet de données. M détecte la présence potentielle d'une boucle, et jette le paquet. Il répond ensuite à S par un message RERR (*Route Error*), pour lui signaler l'erreur de routage. S, recevant ce message d'un de ses prédécesseurs, comprend que celui-ci n'est plus capable d'utiliser la route d'hôte que lui a utilisée. Il supprime donc la route d'hôte décrite à l'intérieur du RERR, et transmet ce message à son propre successeur afin de détruire la route d'hôte jusqu'au puits. Ainsi, le prochain paquet de donnée à destination de F arrivant au puits déclenchera la recherche réactive de F dans le réseau.

Dans le cas (e–h), c'est F qui envoie le paquet de données. M détecte de la même manière la présence potentielle d'une boucle, et rejette de la même manière le paquet. Il répond ensuite à F par un même message RERR, pour lui signaler l'erreur de routage. Contrairement à S dans le cas précédent, F reçoit ce RERR de la part de son successeur, c'est donc que celui-ci n'a pas de route d'hôte dont F est le prochain saut — entre autres pas celle de F lui-même. Il construit donc proactivement sa propre route d'hôte jusqu'au puits à travers son successeur M en lui envoyant un RREP avec un nouveau numéro de séquence. Le prochain paquet de données en provenance de F sera accepté, et cela aura aussi permis de rafraîchir la route d'hôte de F, qui était dans un état instable.

Le RERR ne suit pas exactement le tracé de la route d'hôte, parce que les nœuds ne tracent pas les sauts précédents d'une route d'hôte (on ne peut pas remonter une route d'hôte en sens

inverse) ; mais plutôt, étant envoyé de successeur en successeur, le RERR suit celui de la route par défaut. En général, cela suffit pour suivre une route d'hôte, parce que la méthode de construction d'une route d'hôte la fait justement aller dans le sens inverse de la route par défaut. Il peut arriver, suite à des changements dans l'arbre de collecte, que les deux routes ne soient pas alignées, et que le successeur d'un nœud ne soit pas le saut précédent de l'une de ses routes d'hôtes. Il recevra donc un RERR au sujet d'une route qu'il ne connaît pas. Cette situation n'est pas problématique, car la route d'hôte sera supprimée en partie, au minimum sur le dernier saut, et la suppression reprendra au prochain paquet de donnée envoyé le long de cette route d'hôte obsolète.

2.2 Différence entre LRP avec les protocoles de routage dont il hérite

Cette section présente un aperçu succinct des grandes différences entre LRP et les deux protocoles dont il est inspiré : RPL et AODV/LOADng. Quelques unes de ces différences sont également mises en évidence par les expérimentations décrites dans le chapitre suivant.

Le protocole RPL a été décrit comme très exagérément complexe, comme émettant un large nombre de volumineux paquets de contrôle lorsqu'il n'est pas configuré très précisément [95, 103], en partie à cause de l'instabilité constante de la sélection du parent préféré [96]. Il a également été pointé du doigt en ce qu'il n'atteignait pas les spécifications posées [88, 89]. Le but de cette section n'est pas non plus de décrire ces reproches, mais de comparer les solutions proposées par LRP, RPL et AODV/LOADng aux problèmes posés dans cette thèse.

Plusieurs articles comparent également RPL et LOADng eux-mêmes, et leurs conclusions sont parfois contradictoires [83, 98, 99, 101, 105] ; ce n'est pas très surprenant, car le choix entre un comportement proactif ou réactif est principalement dicté par la densité et la taille du réseau, mais aussi par le modèle de trafic. Le problème consiste donc davantage à évaluer correctement le cas d'utilisation pour ensuite utiliser l'approche appropriée.

Arbre de collecte L'objectif de l'arbre de collecte est d'extraire facilement hors du réseau les données produites par les réseaux de capteurs. LRP et RPL utilisent tout deux un tel arbre [8] et le construisent en utilisant des variantes de l'algorithme de Bellman-Ford distribué, où chaque nœud diffuse sa position en multidiffusant un DIO et choisit la meilleure position qu'il peut avoir dans l'arbre en fonction des DIO reçus de ses voisins. La différence entre eux réside dans la façon dont cet arbre est entretenu.

RPL utilise l'algorithme *Trickle* [30] pour réduire le trafic de routage lorsque la topologie ne change pas. Conformément à cet algorithme, les messages sont transmis plus fréquemment au début et ensuite avec de plus en plus de temps entre les réémissions. D'un côté, les avantages de cet algorithme sur le long terme sont évidents : de moins en moins de paquets sont envoyés. D'un autre côté, avant que les émissions soient très espacées, les nœuds envoient de nombreuses fois des informations redondantes. Cela est particulièrement coûteux pour les couches MAC comme ContikiMAC où le coût énergétique des multidiffusions est généralement d'un ordre de grandeur supérieur à celui des monodiffusions. Dans le cas où le mode de transmission en couche MAC est un mode avec balises (*beacon-enabled* ou TSCH), l'algorithme *Trickle* devient inutile ; mais les DIO doivent alors rester très petits pour pouvoir être transmis avec la balise !

LRP initialise l'arbre de collecte, puis s'arrête complètement d'émettre. L'effet bénéfique est immédiat : les nœuds n'émettent que peu de messages de contrôle. L'inconvénient pourrait venir du fait que les changements de topologie sont moins bien détectés — d'où la nécessité d'être

8. Je rappelle que RPL construit plus exactement un DODAG plutôt qu'un arbre de collecte — mais la façon de le construire et le but sont les mêmes, la seule différence entre eux étant qu'un DODAG tolère qu'un nœud aie plusieurs chemins pour atteindre la racine du graphe, le puits du réseau.

capable d'estimer rapidement la qualité d'un nouveau lien, sujet que nous aborderons dans le chapitre 4 — mais c'est en fait un point moins gênant qu'en première approche : si les liens apparaissants sont ignorés, cela n'empêche pas le réseau de fonctionner ; si les liens disparaissent, ils sont détectés lors de la transmission des paquets de données et LRP réagit alors pour réparer l'arbre.

De son côté, LOADng propose également l'installation d'un arbre de collecte, mais comme une extension au protocole, appelée LOADng-CTP [102]. L'arbre de collecte est enraciné dans un nœud précis : ses branches sont composées de routes d'hôtes pointant sur la racine elle-même, plutôt que des routes par défaut. Un point sur lequel ce protocole insiste beaucoup est la vérification de la qualité des liens avant la construction de l'arbre de collecte, à l'aide d'un message HELLO [29].

Réparation de l'arbre Lorsque la topologie change, le protocole de routage doit corriger les routes choisies, et entre autres l'arbre de routage. Or l'algorithme de Bellman-Ford distribué réagit très mal lorsqu'un nœud s'éloigne de la destination, comme nous l'avons vu, en conduisant dans certains scénarios à un comptage à l'infini entre les nœuds.

Avec RPL, ce problème est résolu en tolérant un début de comptage à l'infini, jusqu'à une valeur maximale fixée par le puits, `DAGMaxRankIncrease`. Les boucles sont donc uniquement transitoires et vite arrêtées. Dans le cas général, cela suffit pour pouvoir trouver une route alternative pour joindre le puits. Dans certains cas particuliers, les nœuds peuvent se trouver incapables de réintégrer l'arbre après avoir reculé autant que cela est toléré.

La maintenance de l'arbre de collecte de LOADng-CTP est accélérée par l'utilisation des Smart-RREQ [93] ; mais cette façon de faire conduit à un blocage dans des cas où une inversion de lien est nécessaire pour atteindre la racine de l'arbre.

LRP garantit de son côté l'absence de boucles dans l'arbre de collecte à tout instant. Ce point est très important, parce qu'il sert comme nous l'avons vu lors de la validation de l'acheminement des données. Pour ce faire, il propose un mécanisme de réparation locale, correspondant à sa logique de réagir à cet événement lors de son apparition, et autant que possible avec un impact localisé.

Entretien des routes d'hôtes Chacun à sa façon, mais selon une logique commune aux protocoles à vecteur de distances, les trois protocoles de routage établissent les routes d'hôtes à l'aide d'un message circulant dans le sens inverse de la route. D'un côté, AODV, de façon réactive, demande la génération de ce message RREP en inondant le réseau d'un message RREQ. De leur côté, RPL et LRP émettent proactivement ce message (resp. un DAO (*Destination Advertisement Object*) et un RREP).

Il arrive cependant qu'une route établie vienne à ne plus être utilisable. Le comportement de LRP se rapproche alors de celui d'AODV en venant réactivement demander à l'hôte de rétablir sa route. RPL, lui, ne fournit aucun moyen au puits d'inciter un nœud à rafraîchir sa route d'hôte [89] (ou alors en incrémentant le DTSN, qui sollicite le réseau entier). Les nœuds sont donc forcés d'émettre des DAO en aveugle à intervalles réguliers, afin d'être sûrs que le puits a toujours moyen de les joindre. La période de renouvellement des DAO est un compromis entre la quantité de trafic de contrôle qu'ils génèrent et le temps pendant lequel un nœud peut rester inaccessible.

Validation de l'acheminement Dans RPL, l'acheminement des données est validé par une option insérée dans leur en-tête IPv6 contenant la direction du paquet et le rang de l'émetteur. De cette façon, chaque paquet envoyé contribue à vérifier la topologie — ce mécanisme est hérité de CTP (*Collection Tree Protocol*) [81]. Malheureusement, cette façon de faire a un prix élevé : pour les paquets provenant de l'extérieur ou à destination de l'extérieur du réseau, comme on ne peut

pas ajouter des en-têtes à un paquet IPv6, il devient obligatoire de les encapsuler dans un autre paquet IPv6, ce qui est d'autant plus gênant que les trames transmises à l'aide de IEEE 802.15.4 sont courtes (127 octets de PDU à la couche MAC).

LRP utilise un mécanisme de validation qui ne nécessite pas l'ajout d'information dans le paquet transmis. Il est basé, comme nous l'avons vu, sur la vérification de la cohérence entre la route suivie et le nœud qui a transmis le paquet au saut précédent.

Du côté d'AODV, la question ne se pose pas : les routes étant toutes disjointes, elles ne peuvent pas interagir, et séquencées, elles ne peuvent pas créer de boucles.

2.3 Notes & recommandations d'implémentations

LRP a été développé sur le système d'exploitation Contiki, comme nous le verrons dans la section 3.1. Le développement de cette implémentation a apporté des leçons intéressantes, que je détaille ici. Les règles régissant les numéros de séquence, bien connues dans la littérature, y sont également abordées.

2.3.1 Écriture des informations dans la mémoire non volatile

Pour garantir la cohérence du réseau, quelques informations ne doivent pas être perdues par les nœuds : leur position dans l'arbre (l'adresse du DODAG, le numéro de séquence de l'arbre et la distance jusqu'au puits), et le numéro de séquence interne utilisé lors de l'émission des RREP. Ces informations doivent être conservées même si un nœud redémarre, il est donc judicieux de stocker ces informations dans la mémoire non volatile, par exemple en utilisant le système de fichiers en mémoire flash Coffee [82].

2.3.2 Inondation vs. multidiffusion

LRP a besoin, à certains moments de multidiffuser un message localement, et à d'autres moments d'inonder tous les nœuds du réseau local d'un message. Aucune méthode spécifique n'est définie pour réaliser ces opérations. La meilleure méthode dépend de la couche liaison de données utilisée, donc de l'implémentation.

Inonder d'un message ne signifie pas que tous les nœuds doivent émettre le message, mais plutôt que tous les nœuds doivent recevoir le message. Les messages DIO, RREQ et BRK sont retransmis par des voisins, ce qui génère des collisions. L'opération de multidiffusion doit traiter avec attention les collisions si la couche liaison de données ne propose pas de mécanisme pour atténuer leur impact, et peut-être ajouter un délai aléatoire lors de la retransmission de paquets multidiffusés.

Quelques nœuds peuvent être configurés pour ne pas router les paquets des autres nœuds, ce qui peut être utile si un nœud a peu d'énergie à sa disposition. Dans ce cas, il ne doit pas diffuser les messages DIO (sauf lorsqu'il sonde son environnement local avec un DIO contenant un numéro de séquence nul), RREQ et BRK. Néanmoins, le nœud doit quand même être capable d'envoyer des messages RREP et de répondre aux messages RREQ et RERR.

2.3.3 Détection de l'inaccessibilité des voisins

Le mécanisme de réparation locale décrit en section 2.1.1.4 démarre lorsque le successeur est déclaré inaccessible. LRP ne définit aucune méthode pour réaliser la détection de l'inaccessibilité des voisins (NUD). Si la couche liaison de données n'est pas capable de détecter l'inaccessibilité d'un voisin (ce qui est souvent le cas lors de l'utilisation des liens sans fil des réseaux de capteurs)

l'implémentation doit produire un moyen de la détecter. Lorsqu'IPv6 est utilisé, l'algorithme de NUD (*neighbor unreachability detection*) d'IPv6 [26] peut par exemple être implémenté. Le protocole BFD (*Bidirectional Forwarding Detection*) propose également « [la détection] de fautes dans le trajet bidirectionnel entre deux relais (*forwarding engines*), [c'est-à-dire dans] les interfaces, [dans] le ou les liens de données, et dans la mesure du possible [dans] les relais eux-mêmes, avec une latence potentiellement très faible », et cela « indépendamment [du] protocole de routage » [27].

2.3.4 Numéros de séquence

Le protocole LRP utilise des numéros de séquence. Le puits maintient celui de l'arbre de collecte, et l'incrémente lors de réparations globales et lors de réparations locales (mais avec une portée restreinte). Il maintient également le numéro de séquence des RREQ émis lors de la recherche réactive des routes. Les nœuds maintiennent aussi leur propre numéro de séquence, utilisé pour la création de routes d'hôtes (messages RREP) ou lors des réparations locales (messages BRK). Si un nœud transmet un tel message, il ne doit pas modifier le numéro de séquence, mais conserver celui de l'émetteur original du message.

Les numéros de séquence sont utilisés dans beaucoup de protocoles. La RFC 1982 [16] définit une arithmétique possible pour les manipuler. Elle donne quelques explications sur leur codage et leur signification :

> « Les numéros de séquence sont formés à partir d'entiers positifs ou nuls d'un sous-ensemble fini des valeurs entières. [...] Lorsqu'on les considère comme un numéro de séquence, aucune valeur n'a une signification particulière, il n'y a pas de numéro de séquence minimum ni de maximum, chaque valeur a un successeur et un prédécesseur. [...] Seulement deux opérations sont définies sur les numéros de séquence, l'addition d'un entier positif sur une plage limitée, et la comparaison avec un autre numéro de séquence. »
>
> — RFC 1982 [16]

L'addition et la comparaison doivent tenir compte de la représentation des numéros de séquence sur un nombre fini d'entiers. Cela implique la gestion des débordements. La RFC définit donc l'addition de deux numéros de séquence s_1 et s_2 représentés respectivement par les entiers non-signés sur N bits en complément à deux i_1 et i_2 en additionnant i_1 et i_2 en éliminant la retenue débordante lorsqu'elle apparaît. La comparaison est faite en comparant les entiers entre eux dans le sens dans lequel ils sont le plus proche : si la distance entre eux est supérieure à la moitié du nombre de valeurs possibles, un débordement est supposé, et une retenue est ajoutée au plus petit des deux nombres pour en faire le plus grand. Formellement,

$$\begin{aligned} s_1 = s_2 &\iff i_1 = i_2 \\ s_1 < s_2 &\iff (i_1 < i_2 \wedge i_2 - i_1 < 2^{N-1}) \vee \\ &\qquad (i_1 > i_2 \wedge i_1 - i_2 > 2^{N-1}) \\ s_1 > s_2 &\iff (i_1 < i_2 \wedge i_2 - i_1 > 2^{N-1}) \vee \\ &\qquad (i_1 > i_2 \wedge i_1 - i_2 < 2^{N-1}) \end{aligned} \tag{2.3}$$

Cette façon de faire permet de comparer pratiquement tous numéros de séquence. Un seul cas particulier arrive, lorsque $|i_1 - i_2| = 2^{N-1}$: on ne sait pas dans quel sens comparer les deux nombres, on ne sait pas s'il faut prendre en compte une retenue. Le résultat d'une telle comparaison reste indéfini dans cette RFC pour des questions de cohérence ; mais les numéros de séquence à comparer restent en général proches, et ce cas est donc assez exceptionnel.

RPL, dans la RFC 6550 [32], s'appuie sur de tels numéros de séquence sur 7 bits et les combine avec une série de numéros de séquence « en sucette » (d'après le terme anglais *lollipop* utilisé par

Perlman [55]). Toutes les valeurs plus petites ou égales à 127 sont utilisées pour l'espace circulaire décrit plus haut ; les valeurs au-delà de 128 (avec le 8e bit de l'octet mis à 1) sont utilisées pour une suite de numéros de séquence linéaire (la tige de la sucette), qui ne boucle pas, afin d'indiquer le redémarrage du compteur : toutes les valeurs supérieures ou égales à 128 décrivent des numéros de séquence plus petits sémantiquement que ceux entre 0 et 127.

LRP, à l'instar de LOADng, utilise des numéros de séquence comme définis dans la RFC 1982 sur 16 bits [9]. En plus de cela, LRP exclus le numéro de séquence 0 des numéros de séquence, afin d'en faire un marqueur. Il est utilisé par les nœuds qui ne se sont jamais associés au réseau, lorsqu'ils émettent un DIO pour sonder leur environnement. Une sorte d'espace en sucette où le bâton de la sucette ne contient que cette valeur-là.

Conclusion

Conçu après de nombreux protocoles de routage, LRP (*Lightweight Routing Protocol*) est spécialement conçu pour offrir un service de routage avec une surcharge faible, afin de répondre spécifiquement aux contraintes inhérentes aux réseaux de capteurs sans fil. Pour cela, il joint la proactivité de RPL à la réactivité d'AODV et de LOADng ; proactivité en ce qu'il construit un arbre de collecte et un jeu de routes d'hôtes toujours prêts à acheminer les données à leur destination ; réactivité en ce qu'il réagit aux changement de topologie à la demande. Les flots de trafic *point-to-multipoint* et *multipoint-to-point* sont transmis le long de l'arbre de collecte, soit depuis le puits soit vers lui — le trafic *point-to-point* est supporté, même si le chemin emprunté par les données n'est alors pas forcément le plus court. Enfin, l'absence de boucle de routage est démontré et assuré en validant le chemin de chaque paquet transmis.

Les mécanismes de LRP et leur raison d'être ont jusque-là été décrits et justifiés théoriquement. Le chapitre suivant présente les résultats de simulations et d'expériences que j'ai réalisé avec ce protocole, afin de vérifier dans la pratique les propriétés décrites par la théorie.

Le travail théorique sur LRP n'en est pour autant pas terminé. Sans même parler de micro-optimisations qui seraient plutôt du ressort de l'implémentation, LRP présente des situations où il est possible de pousser sa légèreté encore plus loin. Un des points où LRP est le plus gourmand en messages émis est lors de l'inondation de RREQ, lors de la recherche d'un hôte dans le réseau. Se passer de ce mécanisme n'est possible que si les nœuds entretiennent proactivement leur route d'hôte — ce qui serait aller à l'encontre de l'approche du protocole. L'idée serait alors d'utiliser la réception d'un paquet de données émis par un hôte du réseau au niveau du protocole de routage, la seule présence de ce paquet indiquant que cet hôte est toujours présent et accessible sur le réseau. Mais, en créant et maintenant ainsi les routes, la validation de l'acheminement doit être regardée de près sous peine de créer des boucles de routage !

9. Il y a bien une toute petite différence : le cas ambigüe où $|i_1 - i_2| = 2^{N-1}$ est défini comme comparable sous LOADng — cependant, comme expliqué plus tôt, ce cas n'arrive jamais dans la pratique, surtout avec des compteurs sur 16 bits.

Chapitre 3

Expérimentations pratiques

Sommaire

Le chapitre précédent décrit et justifie les mécanismes de LRP visant à réduire le nombre de messages de contrôle et garantir la qualité du routage. Ce chapitre considère LRP mis à l'épreuve par la pratique. Au cours de ma thèse, les deux approches ont en fait été réalisées conjointement, la mise en situation de LRP aidant à percevoir les besoins afin de piloter la conception, mais aussi validant les résultats attendus. Ici, les mécanismes de LRP sont observés à l'aide de simulations et d'expérimentations en plateforme réelle. Une comparaison est faite avec RPL. Des expérimentations sont également réalisées dans des situations où l'effet de l'asymétrie des liens sur ces protocoles peut être observé. Les simulations et les expérimentations ne sont pas toutes réalisées avec la dernière version de LRP, d'où quelques différences de comportement du protocole. Ces différences nous permettront d'observer l'effet des mécanismes proposés par LRP et leur intérêt.

3.1 Implémentations disponibles

Deux implémentations de LRP sont disponibles, dans le cadre de ma thèse. La plus complète est l'implémentation faite sous le système d'exploitation pour réseau de capteurs Contiki.

3.1.1 Implémentation dans Contiki

Pour confirmer expérimentalement les raisons d'être de LRP, il a été implémenté par-dessus le système d'exploitation Contiki. Le développement a été initié par les membres de l'équipe où j'ai commencé ma thèse, puis je l'ai grandement repris et développé, entre autres pour qu'il supporte toutes les fonctionnalités développées dans le cadre de ma thèse. Au début de ses développements, seul Contiki-OS existait (en version 2.7 pour les plus anciens développements, puis en version 3.0) ; il n'existe donc pas de portage pour Contiki-NG, mais avec l'essor de celui-ci et le déclin de Contiki-OS, ce serait quelque chose d'intéressant pour la poursuite des expérimentations.

Posséder une implémentation de LRP sur Contiki-OS a aussi l'avantage de pouvoir facilement le comparer avec RPL, puisque celui-ci était déjà implémenté à l'intérieur du système d'exploitation [86].

Toutes les fonctionnalités décrites dans le chapitre 2 sont aujourd'hui supportées par cette implémentation de référence, qui est utilisée lors des expériences présentées dans ce chapitre. Le code est disponible en ligne, dans un dépôt GitHub de l'équipe Drakkar [6] :
https://github.com/drakkar-lig/contiki/tree/lrp/core/net/lrp

3.1.2 PyLRP

Durant ma thèse, j'ai aussi développé une implémentation minimale de LRP en Python sous Linux en espace utilisateur, en utilisant netfilters et iptables [130]. Les tests sont basés sur les conteneurs légers Docker [126], connectés entre eux sur la même machine. Il s'agit principalement d'une preuve de concept, tous les mécanismes ne sont pas implémentés ; mais l'implémentation est suffisamment avancée pour tester son fonctionnement.

Les capteurs ne sont en général pas assez puissants pour faire tourner un noyau Linux et un interpréteur Python. Des appareils connectés peuvent pourtant être compatibles IEEE 802.15.4, sans posséder les contraintes mêmes des capteurs — le meilleur exemple est encore le puits duquel il est attendu, du fait de ses fonctions de routeur de bordure, qu'il soit plus puissant que les autres capteurs du réseau. De plus, sans parler de la norme IEEE 802.15.4, l'implémentation peut servir dans d'autres réseaux, et particulièrement pour accélérer des simulations en exécutant directement le code Python sur le système natif de la machine de simulation plutôt qu'en émulant le code des capteurs, ce qui peut mener à des expérimentations plus complexes et avec beaucoup plus de capteurs.

Le code de ce projet est disponible en ligne, dans un dépôt GitHub [5] :
https://github.com/audeoudh/pylrp

3.2 Expérimentations en simulateur

3.2.1 Simulations avec Cooja

Cooja [73] est un simulateur réseau écrit en Java, développé conjointement à Contiki depuis la version 2.0. Il est capable d'émuler complètement un nœud, en exécutant le code natif compilé pour cette plateforme. Les interactions radio et la propagation des messages à travers le canal radio sont également simulées.

Lors de la simulation, le modèle de canal utilisé est *Unit Disk Graph*, qui suppose que les nœuds émettent un signal radio décodable dans un périmètre bien déterminé autour du nœud, et pas au-delà. Le choix de ce modèle de canal a été fait par recherche de simplicité. Nous verrons au cours de la première expérimentation dans la section suivante, que ce choix masque totalement les faiblesses de la métrique *hop count* que j'utilise.

J'ai utilisé le simulateur Cooja pour évaluer l'efficacité de l'algorithme de réparation locale. La figure 3.1 présente l'emplacement des nœuds pendant la simulation. 34 nœuds y sont placés, certains autour du puits, d'autres sous forme d'un sous-arbre, et commencent tous à fonctionner dès le début de la simulation. Parmi les nœuds, certains sont des nœuds clients qui envoient du trafic vers le puits (trafic ascendant utilisant les routes par défaut) ; celui-ci répond en renvoyant un autre paquet (trafic descendant utilisant les routes d'hôtes). Ces nœuds sont placés comme feuilles du sous-arbre, ce qui nous permet de tester la connectivité des nœuds dans le réseau. Après 1 minute et 40 secondes, l'initialisation du réseau est faite, et les routes (montantes & descendantes) sont établies ; le sous-arbre avec les nœuds clients se déplace alors d'un endroit à un autre, ce qui génère une rupture de lien entre la tête du sous-arbre et le reste du réseau. RPL et LRP sont placés dans ce scénario pour étudier leur comportement. La figure 3.2 montre les émissions de paquets de contrôle (générés par le protocole de routage) pendant les 10 minutes de la simulation.

Au début, les nœuds LRP construisent un arbre de collecte en émettant des messages DIO ; ils établissent également de manière proactive des routes d'hôtes en transmettant des messages RREP. 30 secondes après, les premiers paquets sont transmis, et certains d'entre eux déclenchent l'établissement réactif d'une route d'hôte (émission de messages RREQ et RREP), parce que l'établissement de certaines d'entre elles a échoué en raison de collisions entre les messages RREP (tous les nœuds démarrent en même temps). De son côté, RPL diffuse des messages DIO et DAO pour construire respectivement les routes par défaut et d'hôtes. L'utilisation de l'algorithme *Tri-*

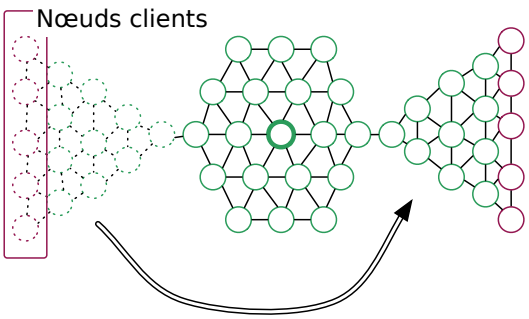


FIGURE 3.1 – Placement des nœuds pendant la simulation avec Cooja. Un sous-arbre entier se déplace à un emplacement différent après 1 minute 40, brisant les routes vers et depuis les clients.

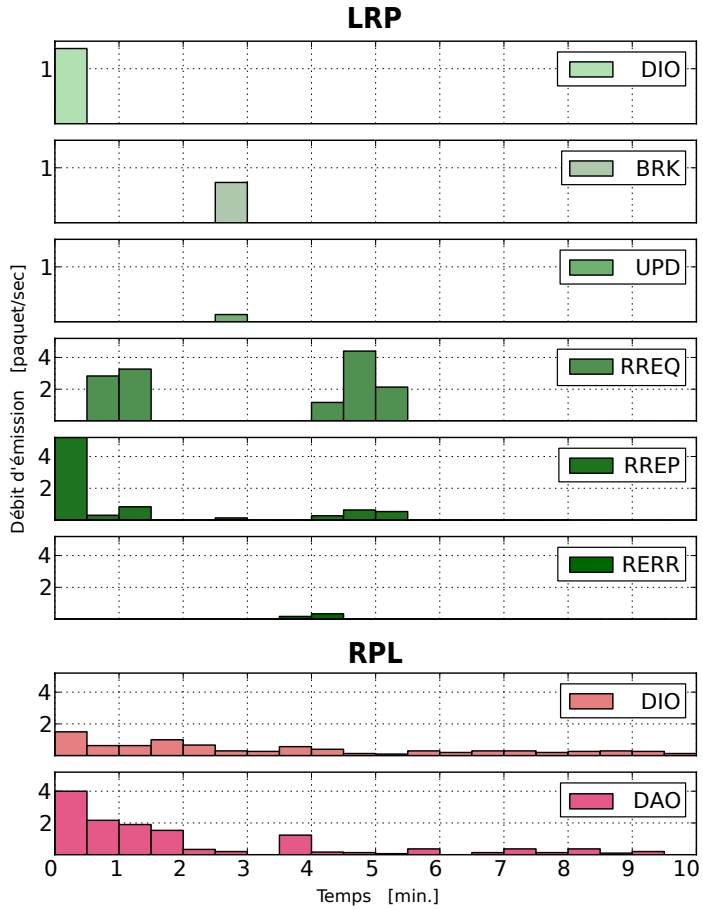


FIGURE 3.2 – Paquets de contrôle générés par RPL et LRP lors de la simulation avec Cooja. Au bout de 1 minute 40, un sous-arbre est déplacé dans le réseau, impliquant la réparation des routes quelques instants après. Les 6 graphiques en haut correspondent aux messages de LRP, les 2 graphiques en bas aux messages de RPL. (Les échelles ne sont pas les mêmes entre les 3 graphiques du haut et les 5 du bas.)

ckle par RPL fait que les messages DIO sont émis en continu durant la simulation, bien qu'ils soient de plus en plus espacés. Les messages DAO continuent aussi à être présents à un rythme moins élevé pour l'entretien de ces routes.

LRP effectue une réparation locale suite au mouvement du sous-arbre, après 1 minute 40 de simulation. La rupture du lien n'est détectée qu'un peu avant la 3e minute, le délai entre les deux étant dû au NUD (*neighbor unreachability detection*). Les nœuds envoient des messages BRK et UPD en appliquant le mécanisme de réparation local et se réassocient à l'arbre de collecte dans leur nouvelle position, mais les routes d'hôtes sont encore dirigées vers l'ancienne position du sous-arbre. Les nœuds étant inaccessibles à cet endroit-là, elles sont supprimées par des messages RERR après un autre délai impliqué par le NUD, et reconstruites de manière réactive grâce aux messages RREQ et RREP.

Il est à noter que, lors de ces simulations, le mécanisme de restauration proactive des routes d'hôtes du sous-arbre (tel que décrit en section 2.1.2.3) n'était pas implémenté. Son objectif est justement de diminuer la grosse quantité de RREQ dont tout le réseau est inondé après la 4e minute, lors de la recréation réactive des routes d'hôtes.

L'effet du déplacement du sous-arbre avec RPL n'est que très légèrement visible, à cause de l'émission constante des messages DIO et DAO, qui entretient proactivement les routes par défaut et d'hôtes, sans gros changement dans leur rythme d'émission.

3.2.2 Nombre d'inversions de liens lors de la réparation locale

Lors d'une réparation locale, le message UPD envoyé pour restaurer le chemin depuis le nœud détaché jusqu'au puits peut retourner certains liens. Julien COUTINHO [104], stagiaire au laboratoire en 2015 avec qui j'ai travaillé, a étudié le nombre de liens retournés auquel on doit s'attendre, en fonction de la densité du réseau. Ses résultats indiquent que très peu de liens sont retournés dans des réseaux denses. Il justifie aussi théoriquement ce phénomène en avançant que, « plus un graphe sera dense, plus les nœuds auront la possibilité de se rattacher à un autre nœud, qui a un rang inférieur ou égal à son ancien prédécesseur. Il n'aura donc pas besoin de faire de BRK et donc il n'y aura pas d'inversions. » C'est donc lorsque les alternatives sont peu nombreuses que le mécanisme de réparation locale a un grand effet, justement quand, sans lui, les nœuds sont susceptibles de créer des boucles de routage ou d'être totalement isolés.

3.3 Expérimentations en plateforme réelle

Pour valider la performance des schémas proposés dans des conditions réelles, nous avons réalisé des expérimentations sur la plateforme d'expérimentations IoT-LAB [106, 127] déployée dans divers centres de recherche en France par l'établissement FIT (*Future Internet Testing*). Cette plateforme propose un banc de test composé de plus de 1500 capteurs sans fil et autres objets communicants déployés à travers six sites en France. Elle permet de pousser les tests plus loin que les simulations en plaçant les protocoles dans un environnement réel, qui prend en compte le comportement vrai des capteurs et des transmissions radio.

Cette section décrit les expérimentations pratiques réalisées pour évaluer la performance des protocoles de routage. Dans la première, les différents mécanismes de LRP sont illustrés. Ensuite, une comparaison est faite au sujet du trafic en arrière plan entre RPL et LRP. Enfin, ces deux protocoles sont étudiés en présence de liens asymétriques.

Pour toutes les expérimentations, nous utilisons les capteurs M3 proposés par IoT-LAB [123], basés sur un processeur ARM Cortex M3. Ils peuvent être utilisés avec Contiki et l'implémentation déjà utilisée pour les simulations. Ils possèdent une interface radio AT86RF231 compatible

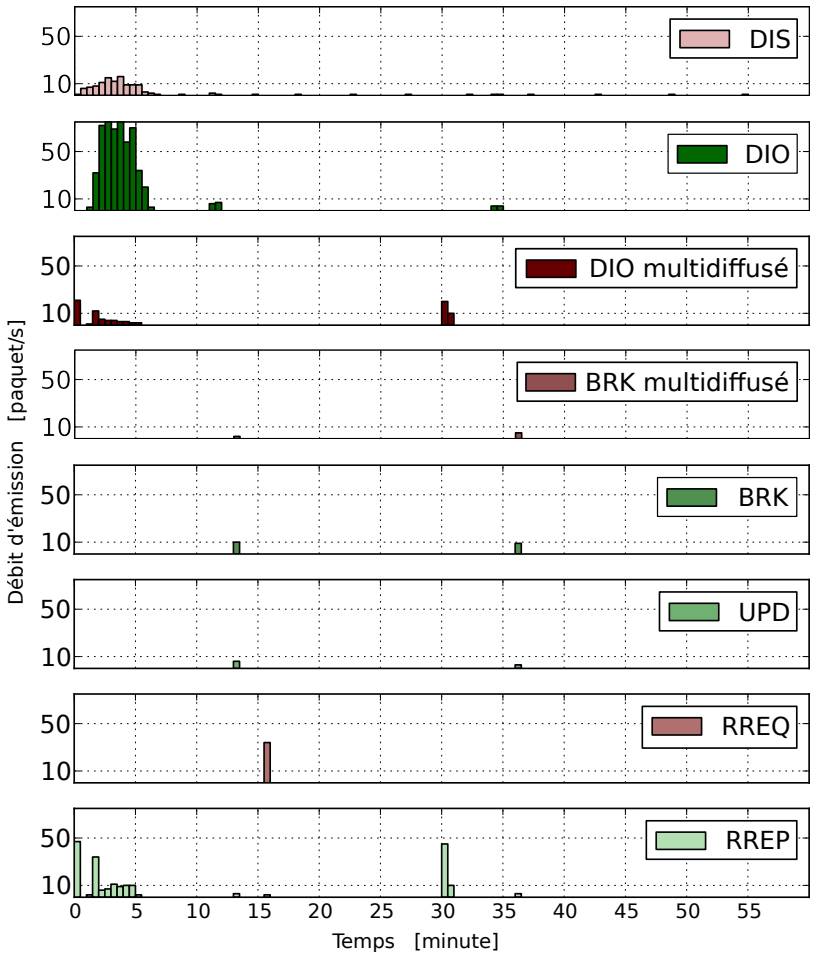


FIGURE 3.3 – Paquets de contrôle générés par LRP lors de l'expérimentation [EXPÉ1]. Une réparation globale (programmée) est initiée par le puits à la 30e minute.

avec IEEE 802.15.4. Ils communiquent, pendant les expérimentations, sur les canaux 11 à 26 de la page de canaux zéro, c'est-à-dire les 16 canaux de la couche physique O-QPSK dans la bande des 2450 MHz. La fréquence centrale est donnée par la formule suivante où k, compris entre 11 à 26, est le numéro du canal :

$$F_c = 2405\,\text{MHz} + (k - 11) \times 5\,\text{MHz}$$

La couche MAC utilisée est ContikiMAC, nativement implémentée dans Contiki. Les nœuds communiquent entre eux en utilisant IPv6 et la couche 6LoWPAN d'adaptation à IEEE 802.15.4.

3.3. [EXPÉ1] Observation des mécanismes de LRP

Dans cette expérimentation, 35 nœuds, tous placés sur une grille 3D d'IoT-LAB, utilisent l'algorithme de routage LRP. Le puits a aussi le rôle de serveur : chaque client envoie un message toutes les 15 secondes (simulant du trafic collecté hors du réseau) auquel le serveur répond (trafic distribué au-dedans du réseau). Le trafic des messages de contrôle (générés par le protocole de routage) est présenté dans la figure 3.3.

Cette expérimentation illustre bien le fonctionnement des mécanismes de LRP. Ici, LRP est encore dans une version antérieure à celle décrite dans le chapitre 2 : lorsqu'ils sont déconnectés, les nœuds émettent des messages DIS dont le rôle est le même que ceux de RPL : solliciter des messages DIO monodiffusion de leurs voisins. Ces messages ont été remplacés depuis par l'émis-

sion de messages DIO indiquant la position du nœud, potentiellement infinie (cf. détails dans la section 2.1.1.2).

Ces messages DIS sont bien visibles à l'initialisation du réseau, où tous les nœuds sont encore dissociés du réseau. La réponse leur parvient de la part des voisins déjà associés (les DIO monodiffusés) et, une fois associés, ils annoncent leur position (DIO multidiffusés). Ils construisent aussi leur route d'hôte de façon proactive (RREP transmis jusqu'au puits) ; ainsi, il n'y a au départ aucun RREQ. À la 16e minute, nous observons toutefois la recherche réactive d'un hôte, sa route n'ayant pas été construite complètement jusqu'au puits (RREQ et réponses RREP).

Après 30 minutes, une réparation globale programmée dans l'expérimentation a lieu, initiée par le puits qui incrémente le numéro de séquence de l'arbre de collecte. Il n'y a pas de DIS, les nœuds diffusent simplement le nouveau numéro de séquence (via les DIO). Certains nœuds changent de successeur et rafraichissent leur route d'hôte (RREP). Le choix d'une période de 30 minutes est purement illustrative et peut être beaucoup plus longue selon la stabilité des liens, ou le nombre de réparations locales observées par le puits, ou quelqu'autre critère.

Aux alentours de la 11e et de la 34e minute, un nœud qui a perdu son successeur commence par rechercher un successeur alternatif (DIS), et des réponses lui parviennent (DIO monodiffusés). Dans les deux cas cette recherche n'aboutit pas ; le nœud amorce alors une réparation locale (BRK multidiffusés puis monodiffusés jusqu'au puits ; réponses UPD). Une fois les routes d'hôtes reconstruites proactivement, le protocole de routage redevient silencieux. On observe bien que LRP, à part à ces moments où un évènement doit être traité réactivement, demeure silencieux. Il n'y a pas de trafic de contrôle en arrière-plan.

Les paquets DIS qui apparaissent régulièrement correspondent à un nœud qui est hors de portée du reste du réseau. Il s'agit d'un muet total : aucun autre nœud ne l'entend ni ne répond, il reste dissocié du réseau tout au long de l'expérimentation.

On remarque la double réparation locale qui a peut-être même eu lieu sur un seul et unique nœud. La métrique utilisée ici est *hop count*, et cela y joue sûrement pour quelque chose. Cette métrique, déjà utilisée lors des simulations, s'était révélée plutôt efficace ; mais la simulation utilise le modèle de canal *Unit Disk Graph*, qui ne prend pas en compte toutes les caractéristiques de la transmission. Avec de vraies liaisons radio, la métrique *hop count* a tendance à privilégier les liens les plus longs pour atteindre le plus rapidement la destination ; mais lorsqu'ils sont trop longs, ces liens sont également instables, permettant peut-être de détecter le nœud à l'autre extrémité, mais ne permettant pas un échange conséquent de paquets. Cette observation est l'une de celles qui nous ont poussé à regarder de plus près la métrique utilisée par l'algorithme de routage, ce que je détaillerai dans le chapitre 4.

3.3. [EXPÉ2] Étude du trafic en arrière plan

L'objectif de ces expérimentations est de comparer RPL et LRP dans une situation réelle, mais sans situation exceptionnelle, afin d'étudier leur fonctionnement naturel en régime de croisière. La version de LRP utilisée est celle décrite dans le chapitre 2 ; mais également une autre version (nommée ici *o-LRP*, LRP original, pour la distinguer de la version du chapitre 2, nommée LRP), où plusieurs fonctionnalités ne sont pas activées, afin de vérifier l'efficacité de ces mécanismes. Il s'agit des mécanismes de DIO sollicités (cf. section 2.1.1.2), de la gestion des liens asymétriques via le message HELLO (cf. section 2.1.1.5), de la ERS lors de la réparation locale (cf. section 2.1.1.4), et des RREQ confinés (cf. section 2.1.2.3).

Les expérimentations [EXPÉ2] se déroulent ainsi. 41 nœuds de la plateforme IoT-LAB sont utilisés pendant deux heures, et font tourner le protocole de routage testé. Les nœuds envoient des paquets UDP de données jusqu'au puits avec une période de 5 minutes, et le puits répond

TABLE 3.1 – Paramètres des nœuds lors des expérimentations [EXPÉ2] et [EXPÉ3].

Paramètre	Valeur
Check interval de ContikiMAC	125 ms
Nombre de paquets UDP par client	1 toutes les 5 min
RPL	
$I_{\min}$	4 s
$I_{\max}$	1048 s
Redondance des DIO	10
Période de régénération des DAO	15 à 22 min
Fonction objectif	MRHOF avec ETX
LRP	
Délai aléatoire pour la transmission d'un paquet lors d'une inondation	0,5 s
Fréquence de sondage du voisinage lorsque le nœud est déconnecté	~5 min
Durée de mise en liste noire	10 min
Type de métrique	*hop count*

TABLE 3.2 – Nombre de messages envoyés lors des expérimentations [EXPÉ2]. La durée de l'expérimentation est de deux heures.

Type de trafic	Diffusion	RPL	o-LRP	LRP
Maintenance de l'arbre de collecte	multi–	786	136	155
	mono–	2497	22	399
Maintenance des routes d'hôtes	multi–	—	0	8
	mono–	1586	172	233
Paquets acheminés avec succès		1795 (97,2 %)	1836 (98,9 %)	1819 (98,0 %)

à chaque paquet. Le tableau 3.1 présente les paramètres utilisés dans ces expérimentations. Les réglages de l'algorithme *Trickle* correspondent à la pratique recommandée par sa RFC [30], avec $I_{\min}$ quelque peu plus grande que la latence en pire cas de la couche liaison de données [1] (125 ms dans ContikiMAC). Ce paramètre permet d'éviter efficacement les collisions entre les DIO lors de leur pic d'émission. Pour la même raison, les transmissions des DIO, BRK, HELLO et RREQ dans LRP lors d'inondation sont randomisées. La figure 3.4 présente l'arbre de collecte typique obtenu (il change seulement légèrement suivant le protocole). Les émissions de paquets de contrôle et de données sont enregistrées, comptées dans le tableau 3.2 et présentées dans la figure 3.5.

Au début de l'expérimentation, le nombre de messages est similaire pour les deux versions de LRP, mais ils sont de natures différentes. Dans o-LRP, les liens sont utilisés sans avoir été vérifiés. Par chance, dans ce cas, il n'y a pas de lien comme révélé défectueux. LRP les détecte pendant la

1. Ce sont les réglages adaptés de l'implémentation ContikiRPL. La RFC 6206 signale explicitement que « la valeur par défaut de $I_{\min}$ DEVRAIT être spécifiée en fonction de la latence en pire cas d'une transmission à la couche liaison », ce que la RFC 6550 de RPL ne fait pas. Elle définit $I_{\min}$ à 8 ms, une valeur absolument pas satisfaisant pour ContikiMAC et sa durée de diffusion trente fois plus élevée que cette valeur. Cette valeur n'est d'ailleurs pas adaptée à une couche MAC quelconque utilisant IEEE 802.15.4, la durée de transmission des 10 paquets nécessaire à la redondance étant au grand minimum de 20 ms.

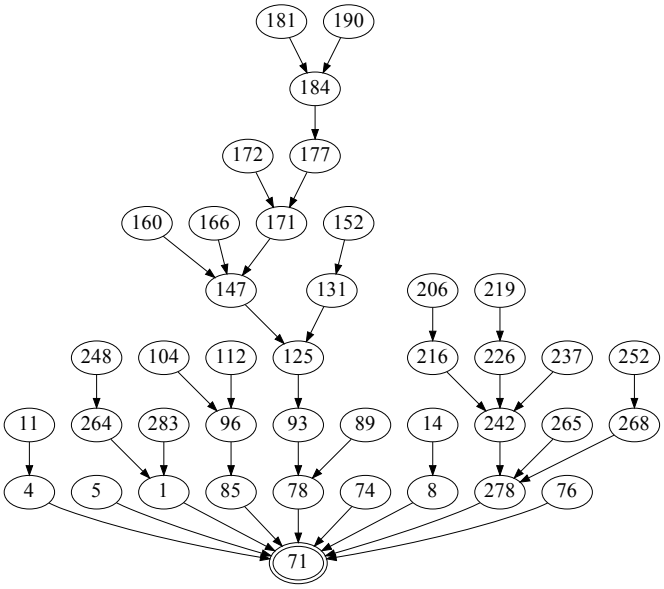


FIGURE 3.4 – Arbre de collecte des expérimentations [EXPÉ2]. Le nœud 71 est le puits. Les arbres sont similaires avec RPL, o-LRP et LRP.

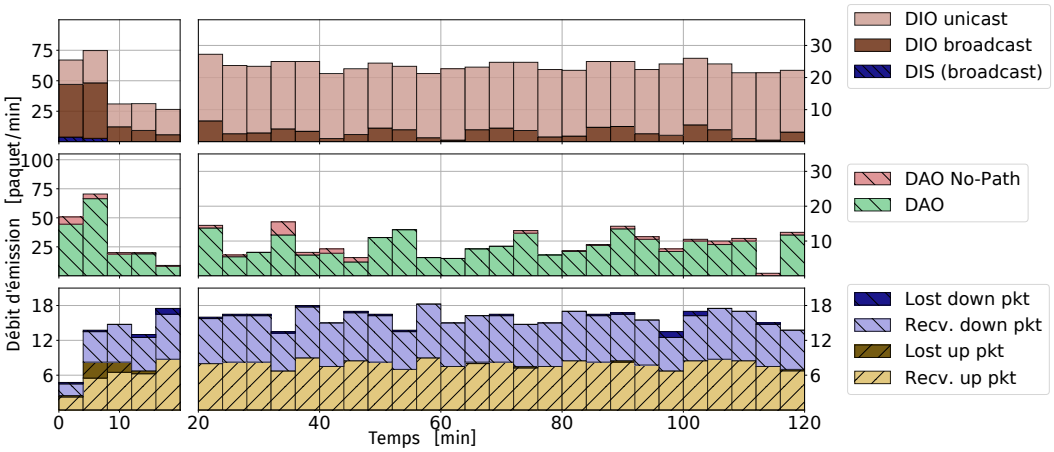
poignée de main HELLO qui ajoute un peu de trafic monodiffusé.

Les trois protocoles de routage acheminent presque tous les paquets de données avec succès (ligne inférieure dans la figure 3.5). Cependant la façon dont ils arrivent à ce résultat n'est pas la même. Tout au long de l'expérimentation, RPL envoie un nombre relativement élevé de DIO multidiffusés (entretien de l'arbre de collecte), monodiffusés (sondage des liens vers les voisins connus, pour entretenir la métrique ETX) et de DAO (entretien des routes d'hôtes). De leur côté, une fois que les routes se sont stabilisées, aucune des deux versions de LRP n'envoie de trafic de routage, sauf lorsqu'un lien est finalement considéré comme inutilisable (p. ex. autour de la 80^{e} minute de l'expérimentation avec LRP). La différence n'est pas si surprenante, il s'agit simplement de la différence entre l'approche proactive et l'approche réactive.

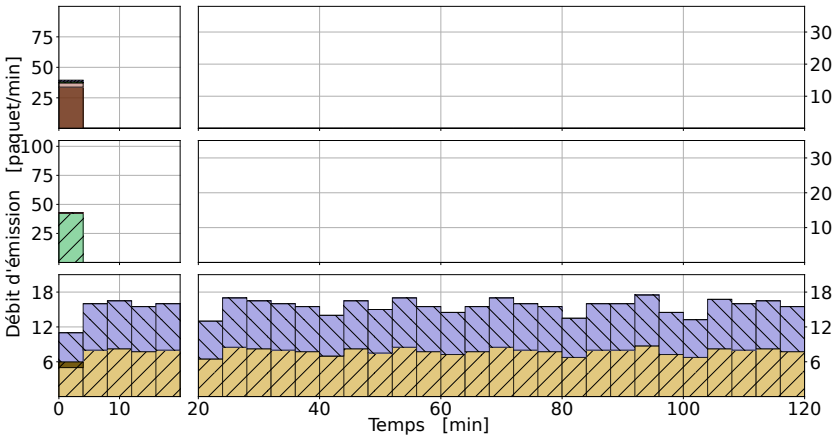
Le trafic de contrôle généré par RPL est donc beaucoup plus grand pour le même résultat. Plus encore, si on considère qu'un paquet multidiffusé a un impact plus fort sur l'occupation du canal qu'un paquet monodiffusé (ce qui est intrinsèque à ContikiMAC ou à la plupart des modes de transmissions CSL) d'un facteur 10, on constate que l'occupation du canal radio avec RPL est plus que cinq fois celle de LRP et plus que deux fois celle de o-LRP dans ce scénario. Ces chiffres ne sont représentatifs que pour ce scénario de deux heures d'expérimentation : si l'expérimentation avait duré plus longtemps, la différence n'aurait fait que s'élargir, à cause du trafic continu de RPL en arrière-plan.

3.3. [EXPÉ3] En présence de liens asymétriques

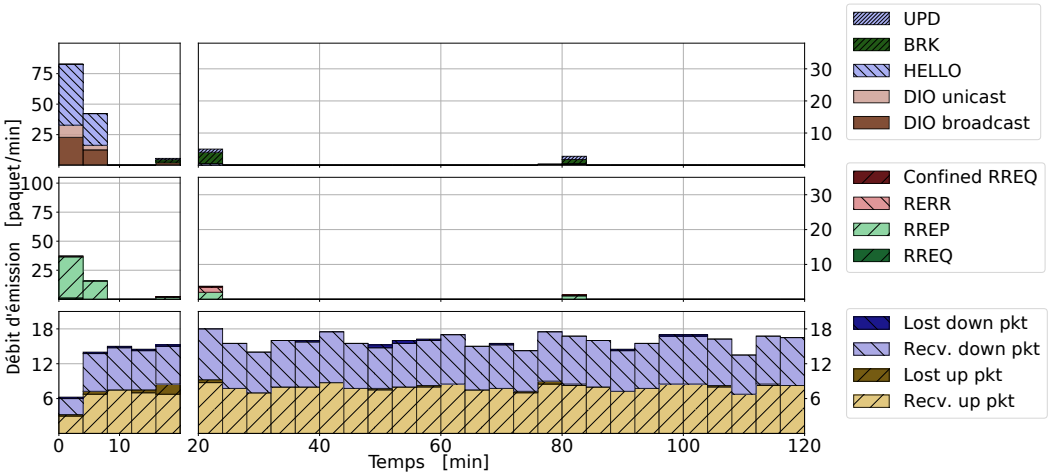
La qualité des transmissions radio dépend de l'environnement et du matériel radio et comme nous l'avons vu dans la section 1.4.2, l'asymétrie a plus tendance à apparaître dans les réseaux de capteurs sans fil où les appareils sont peu coûteux et l'environnement radio peut être encombré. Certains nœuds peuvent alors être sourds (c'est-à-dire que les voisins reçoivent les paquets que ce nœud émet, mais pas l'inverse) ou muets (c'est-à-dire qu'un nœud reçoit correctement, mais n'est pas entendu par son voisinage), avec un degré plus ou moins absolu. Philip LEVIS, co-auteur de RPL, reconnaît lui-même la présence possible de ces liens asymétriques [85]. Les expérimentations [EXPÉ3s] et [EXPÉ3m] permettent d'observer le comportement des protocoles de routage en présence de ces liens.



(a) RPL



(b) o-LRP



(c) LRP

FIGURE 3.5 – Paquets de contrôle générés par les protocoles de routage lors des expérimentations [EXPÉ2].

Ces expérimentations sont assez similaires dans leur mise en place à [EXPÉ2]. Toujours sur la plateforme IoT-LAB et avec ses capteurs M3, les mêmes protocoles de routage sont testés (RPL, LRP et o-LRP), et les mêmes paramètres d'expérimentation sont utilisés (cf. tableau 3.1). Chaque client envoie un paquet UDP toutes les 5 minutes au puits qui répond avec un autre paquet UDP, simulant du trafic montant et descendant. Les réseaux sont toutefois moins peuplés, avec seulement 11 nœuds clients plus un puits, pendant une heure. De plus, l'un des nœuds est configuré pour afficher les propriétés de surdité ou de mutité désirées pour les tests.

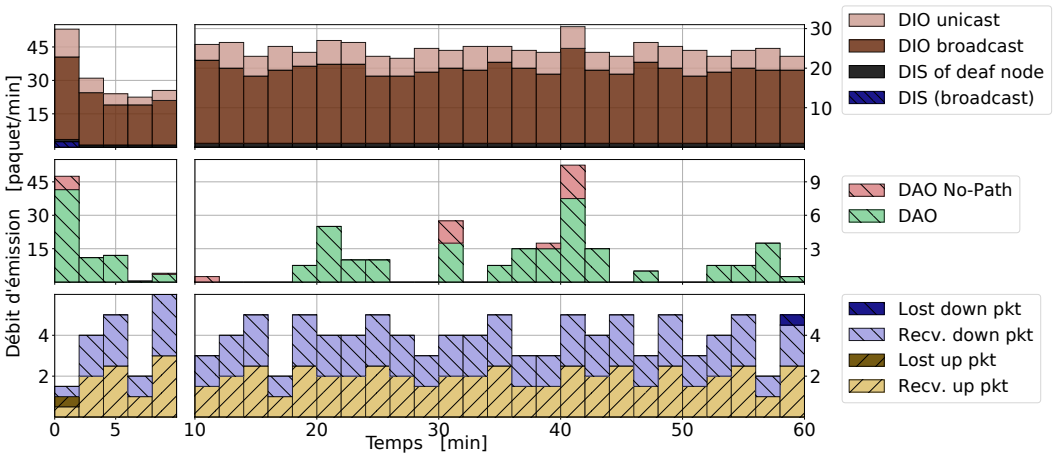
Nœud sourd Dans l'expérimentation [EXPÉ3s], l'un des nœuds clients est sourd : sa sensibilité est réduite et il ne reçoit aucun message des autres nœuds, tandis que ses paquets sont reçus par ses voisins. La figure 3.6 présente les résultats obtenus avec les trois protocoles de routage.

Dans un réseau RPL Lorsqu'un nœud RPL n'est associé à aucun DODAG, il diffuse un message DIS. Ce message réinitialise la minuterie de l'algorithme *Trickle* qui programme les émissions du message DIO dans son voisinage, générant ainsi de nombreuses diffusions DIO. Bien que ce mécanisme soit utile pour accélérer l'insertion de nouveaux nœuds dans le réseau, il est très nuisible lorsqu'un nœud N est sourd. En effet, comme N continue à diffuser sans arrêt des paquets DIS, ses voisins renvoient constamment des DIO multidiffusés, maintenant l'algorithme *Trickle* à son rythme d'émission maximal. C'est ce que nous observons dans la figure 3.6a. Pendant toute la durée de l'expérimentation, les DIO sont constamment diffusés au rythme moyen de 1 toutes les 3 secondes dans cette configuration.

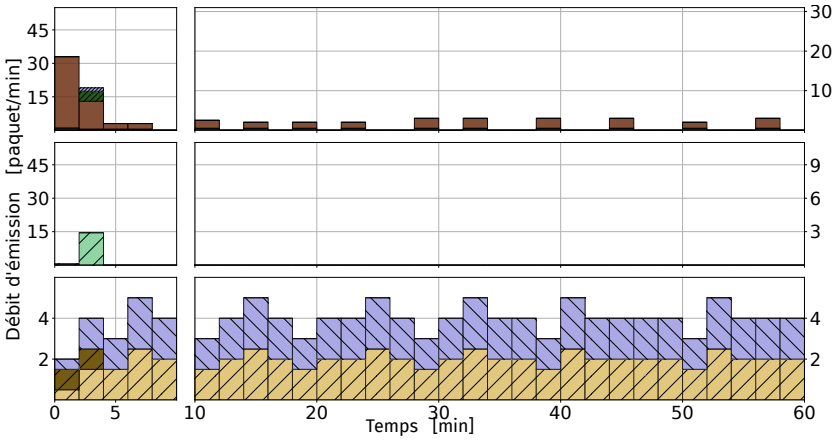
Dans un réseau o-LRP Un peu à l'instar de RPL, le nœud qui n'est pas associé au réseau diffuse sans arrêt des messages DIO pour sonder son environnement. Ce message sollicite une réponse de la part de ses voisins — réponse envoyée en multidiffusion à cause des fonctionnalités enlevées dans o-LRP. Le rythme d'émission des DIO est pourtant moins élevé, parce que RPL utilise l'algorithme *Trickle* qui empire les choses. Nous voyons bien, dans la figure 3.6b, ces réponses envoyées par tous ses voisins à chaque fois que le nœud sourd émet. Ceci conduit à une quantité d'une dizaine de messages multidiffusés toutes les 5 minutes dans cette configuration.

Dans un réseau LRP Les nœuds LRP répondent eux aussi au nœud sourd, mais en utilisant des messages HELLO, pour sonder tout d'abord le lien entre eux et lui, et déterminer conjointement la valeur de la métrique indiquant la qualité du lien entre eux. Comme ils détectent l'asymétrie du lien et la surdité du nœud à cette étape, ils ne répondent pas systématiquement à ses sondes. La figure 3.6c nous montre exactement ce comportement. Le DIO du nœud sourd reçoit seulement par moments des réponses HELLO. Dans cette configuration, LRP envoie un paquet monodiffusé de temps en temps, là où RPL émet un paquet multidiffusé toutes les 3 secondes.

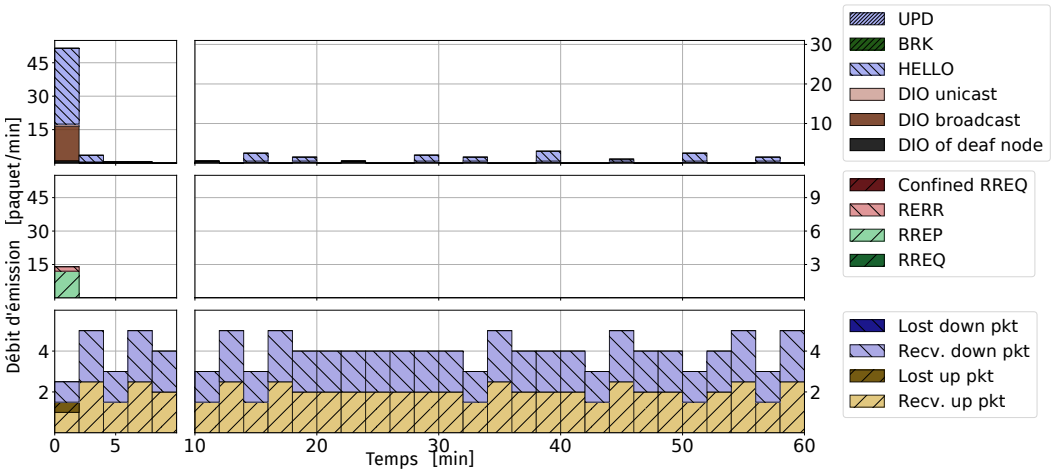
Nœud muet Dans l'expérimentation [EXPÉ3m], l'un des nœuds clients devient muet : sa puissance d'émission est réduite, et les nœuds voisins ne reçoivent pratiquement plus ses messages. Évidemment, si nous coupions complètement les émissions d'un nœud dès le début, il ne dérangerait pas beaucoup ses voisins car tous ses messages seraient perdus et son existence même serait ignorée. On démarre donc le nœud 40 comme un nœud normal et on le laisse s'associer avec le réseau et communiquer avec les autres nœuds. Puis, après 15 minutes, on réduit sa puissance de transmission : il devient muet et ne peut plus atteindre ses anciens voisins — à l'exception d'un d'eux, le nœud 33, afin de ne pas être totalement isolé du reste du réseau. Les arbres de collection de RPL et de LRP sont pratiquement les mêmes et sont présentés dans la figure 3.7 ; la position dans l'arbre du nœud 40 avant et après qu'il soit devenu muet y est indiquée. On remarque que le lien entre 33 et 40 est renversé. La figure 3.8 montre les messages de contrôle échangés pendant l'expérimentation. Le tableau 3.3 recense leur nombre entre la 15^{e} et la 25^{e} minute, juste après



(a) RPL



(b) o-LRP



(c) LRP

FIGURE 3.6 – Paquets de contrôle générés par les protocoles de routage lors des expérimentations [EXPÉ3S] (l'un des nœuds clients est sourd).

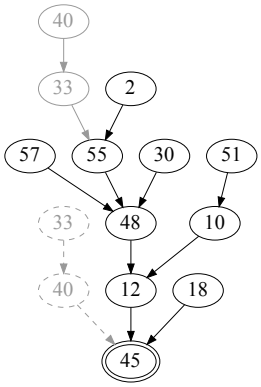


FIGURE 3.7 – Arbre de collecte typique des expérimentations [EXPÉ3m]. Le nœud 45 est le puits. Lors de la construction initiale de l'arbre de collecte, les nœuds 40 et 33 (en pointillés) y sont connectés directement. Après que le nœud 40 soit devenu muet, les deux nœuds ne peuvent se rattacher au réseau que par le nœud 55, et le lien entre eux deux est renversé.

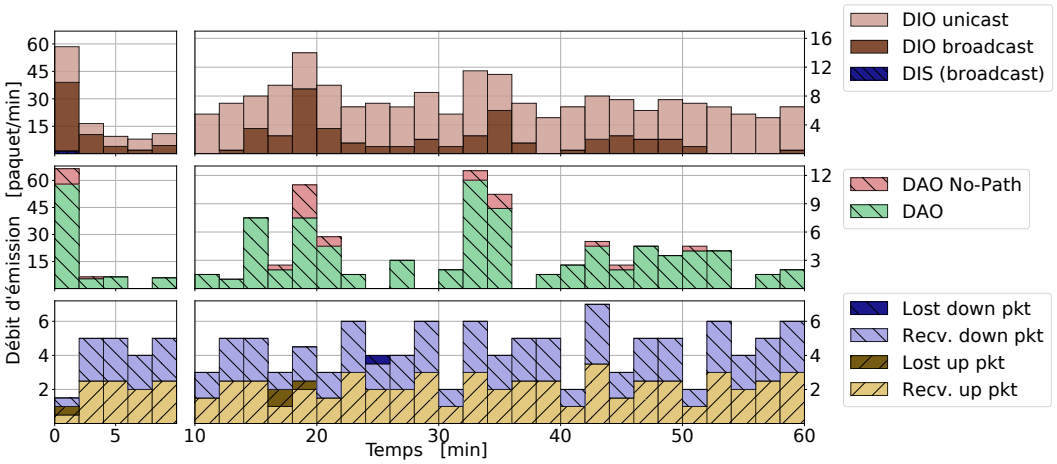
que le nœud 40 soit devenu sourd, au moment où les algorithmes de routage doivent réparer la connexion de ce nœud.

Dans un réseau RPL Le changement de topologie est géré par une boucle transitoire dans le réseau (40 utilise 33 comme successeur, qui à son tour utilise 40 comme successeur, mais le trafic ne fait pas de boucle, grâce à l'en-tête de validation de l'acheminement), jusqu'à ce que les nœuds 33 et 40 soient suffisamment loin du puits pour que 33 puisse s'accrocher derrière 55. Tout ceci génère du trafic DIO supplémentaire entre la 18^{e} et la 21^{e} minute de l'expérimentation (visible dans la figure 3.8a). Cette expérimentation montre que RPL finit par gérer correctement cette situation : peu de paquets de données sont perdus et le DODAG est réparé rapidement.

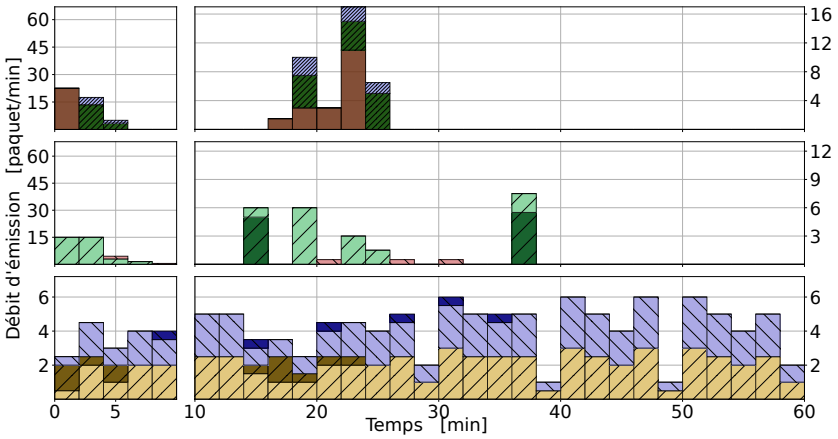
Dans un réseau o-LRP Le nœud 40, n'ayant plus moyen d'atteindre le puits, applique l'algorithme de réparation locale, d'où l'apparition de messages DIO, puis BRK et UPD, peu après 15 min. Cette expérimentation ne se déroule pas exceptionnellement bien. En effet, un phénomène de saturation du réseau arrive, à cause d'un autre nœud qui n'avait pas réussi à construire sa route d'hôte proactivement, d'où une inondation de RREQ qui arrive presque en même temps que la réparation locale dont nous parlons. Cette accumulation de messages multidiffusés (qui sollicitent beaucoup le canal avec ContikiMAC) font que les autres nœuds ont du mal à communiquer et croient observer une dégradation des liens. Ils sondent alors d'autres liens et changent de position, ce qui complique la situation. Celle-ci finit par se résoudre complètement, et le réseau retourne dans un état stable où tous les paquets de données sont acheminés avec succès.

Dans un réseau LRP LRP effectue la réparation locale de la même façon que o-LRP. Il répare également l'ensemble des routes d'hôtes de façon proactive avec le mécanisme des RREQ confinés.

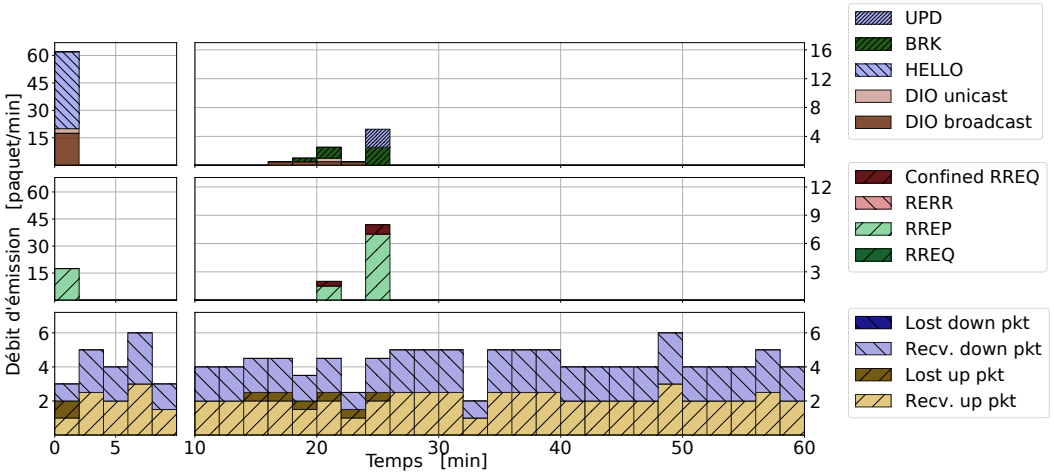
Dans les trois expérimentations, la connectivité est rétablie en quelques minutes. Cependant, comme le montre le tableau 3.3, LRP utilise un cinquième des messages diffusés par rapport à RPL, et quelques messages monodiffusion (9 fois moins que RPL). C'est un autre exemple de la consommation plus élevée de trafic de contrôle nécessaire pour maintenir les routes descendantes de façon proactive.



(a) RPL



(b) o-LRP



(c) LRP

FIGURE 3.8 – Paquets de contrôle générés par les protocoles de routage lors des expérimentations [EXPÉ3m] (l'un des nœuds devient muet après 15 min).

TABLE 3.3 – Nombre de messages de contrôle envoyés lors des expérimentations [EXPÉ3m] entre 15 et 25 min.

Message	Diffusion	RPL	o-LRP	LRP
Maintenance de l'arbre de collecte				
HELLO	mono–	—	—	2
DIO	multi–	66	37	4
DIO	mono–	146	—	1
BRK	multi–	—	7	5
BRK	mono–	—	20	4
UPD	mono–	—	12	5
Maintenance des routes d'hôtes				
DAO	mono–	95	—	—
DAO No-Path	mono–	15	—	—
RREQ	multi–	—	11	3
RREP	mono–	—	25	17
RERR	mono–	—	3	0

Conclusion

Ces expérimentations permettent de consolider l'approche théorique que nous avons eue au chapitre 2. Placé dans un contexte de simulation et d'expérimentation, LRP se révèle particulièrement silencieux, le nombre de messages de contrôle étant suffisant pour maintenir le réseau en fonctionnement. Le service de routage est ainsi offert de façon fiable, permettant aux capteurs de produire du trafic applicatif à destination de l'extérieur du réseau et d'en recevoir depuis l'extérieur. À l'inverse, RPL, alors même qu'il est conçu pour les réseaux LLN à faible puissance, se révèle émettre un grand nombre de messages pour obtenir la même qualité de service.

Deuxième partie

Qualité des liens radio

Chapitre 4

Estimation fiable et rapide du taux de perte d'un lien radio

Sommaire

L'estimation de la qualité des liens est une question souvent laissée de côté par les protocoles de routage. En effet, c'est une question séparée du routage au sens strict. Elle fait l'objet de moins d'attention au niveau algorithmique et au niveau de l'interopérabilité. Cependant, ce n'est pas une question négligeable. Un bon protocole de routage doit finalement toujours y répondre correctement : la précision des décisions de routage est d'avance biaisée si la qualité du lien radio est mal estimée.

Dans des environnements plus stables, les métriques sont parfois considérées suffisamment constantes, et ne sont mesurées qu'à certains moments, à l'initialisation du lien, ou lorsque de grands changments sont détectés sur le lien (e.g. la métrique d'EIGRP). À l'inverse, dans l'environnement spécifique les réseaux de capteurs sans fil, c'est un travail à effectuer en continue malgré les contraintes intrinsèques à ce type de réseau (faible capacité de calcul, restrictions énergétiques…), car la qualité des liens du réseau peut changer très rapidement et est difficile à prévoir de façon fiable. Selon BACCOUR et al. [91], les qualités essentielles d'une métrique efficace seraient :

l'efficacité énergétique : le calcul de la métrique elle-même doit générer peu de surcharge de calcul ou de communication.

l'exactitude : la métrique doit représenter le comportement réel du lien, afin d'avoir des bases solides pour les décisions de routage.

la réactivité : la métrique doit pouvoir rapidement traduire les changements persistants de qualité des liens, afin que les décisions de routage puissent suivre rapidement les évolutions de la topologie du réseau.

la stabilité : la métrique doit pouvoir résister aux variations temporaires de la qualité du lien, afin d'éviter des changements constants dans les décisions de routage.

Plusieurs méthodes existent pour estimer la qualité des liens radio. Elles peuvent être regroupées en deux grandes familles. Tout d'abord les **estimateurs physiques** qui exploitent des mesures effectuées par le matériel, au moment de la réception d'un paquet ou à n'importe quel autre moment. Parmi d'autres peuvent être cités le RSSI et le LQI, défini par la norme IEEE 802.15.4. Ces mesures sont simples à récupérer et directement obtenues après réception du paquet — donc efficaces énergétiquement. Toutefois, ils fonctionnent à l'échelle d'un paquet seul, et non d'un lien, d'où une réactivité excessive (et une stabilité faible). En outre, la valeur remontée par le matériel étant plus ou moins liée au matériel lui-même, peut être difficile à exploiter et pas forcément portable d'un capteur à l'autre — c'est le cas par exemple du LQI (cf. section 4.2).

Ensuite, il y a les **estimateurs statistiques** [1]. Ils utilisent une approche statistique pour décrire le coût général du lien, et utilisent les valeurs remontées par la radio, la couche physique ou la couche liaison de données. À titre d'exemple, ETX indique le nombre (statistique) de retransmissions attendues d'un paquet avant qu'il soit reçu et acquitté par le récepteur. Ces estimateurs, afin d'être exacts, ont besoin d'un certain nombre de paquets pour converger vers une valeur représentative — sans quoi, l'intervalle de confiance de l'estimateur statistique demeure trop grand. Ce prérequis est réaliste pour des liens très utilisés, où les échanges de données sont nombreux : il est possible d'extraire passivement ces méta-données sans surcharge particulière. Ce prérequis est cependant contraignant pour des liens peu utilisés qui nécessitent une mesure active en générant un trafic artificiel et consommateur en ressources, et à plus forte raison encore pour des liens très récemment découverts où quasi aucun message n'a été échangé.

1. BACCOUR et al. [91] nomme ces deux estimateurs des estimateurs basés sur le matériel (*hardware-based*) et basés sur le logiciel (*software-based*). Cette appellation traduit plutôt la façon dont l'information est calculée, et non pas la façon dont l'information est obtenue — d'où l'utilisation de termes légèrement différents ici.

Le travail présenté ici porte justement sur ce dernier scénario. Il faut un estimateur de la qualité des liens qui soit à la fois fiable et rapide : fiable afin de pouvoir prendre de bonnes décisions de routage au sujet de ce lien, et rapide pour construire cette estimation en un nombre très restreint de paquets, sur des liens peu utilisés ou très récemment découverts. Le PER sera utilisé comme indicateur de la qualité du lien. Les sections suivantes présentent trois approches pour estimer le PER : à partir du RSSI, c'est-à-dire de la puissance du signal reçu (section 4.1), à partir du LQI, l'indicateur de qualité fourni par le matériel (section 4.2). Ces deux approches ont été présentés à la conférence WONS en 2018 [4]. Enfin, la dernière section présente la façon la plus poussée et efficace d'estimer le PER, à partir d'une modélisation plus complexe prenant en compte la puissance du signal et le bruit statistique sur le canal (section 4.3). Cette dernière partie du travail est en cours de soumission.

Description du banc de test expérimental

Cette étude est réalisée expérimentalement sur la plateforme de tests IoT-LAB [127] et avec ses capteurs M3, déjà utilisés dans le chapitre 3. La plateforme d'IoT-LAB est déployée dans divers centres de recherche en France [2], où de nombreuses interactions entre les technologies sur la bande ISM sont attendues, entre autres avec la technologie IEEE 802.11. Le niveau de bruit n'est pas le même d'un site à l'autre, comme nous le verrons dans la section suivante, à cause de la proximité variable de la plateforme avec les stations de travail des utilisateurs du site ou des points d'accès IEEE 802.11. Sur ces plateformes, je mesure expérimentalement le taux de réception des trames sur des centaines de liaisons sans fil entre les capteurs, et relève également les paramètres physiques qui permettront les études présentées ici.

Quelques caractéristiques des sites et de configuration des capteurs sont données dans le tableau 4.1. Comme dans les expériences du chapitre 3, les capteurs utilisent la couche physique O-QPSK dans la bande des 2450 MHz, (les canaux fréquentiels 11 à 26). À cause de la petite taille du site de Lyon, la puissance d'émission est réduite pour éviter de n'avoir que des excellents liens. Pendant l'expérience, les capteurs envoient des trames multidiffusées (*broadcast*), reçues par les capteurs aux alentours. Il n'y a pas de mécanisme d'économie d'énergie activé pour simplifier la synchronisation des capteurs. Tous les capteurs sont commandés via le câble série fournie par la plateforme afin d'éviter les collisions entre les trames de mesure. Ce câble série sert également à remonter les informations relevées par les capteurs — trames reçues et mesures radio. Le CCA (*Clear Channel Assessment*) a été désactivé pour éviter les perturbations ; son effet est discuté et étudié en section 4.3.3.4.

Pour s'assurer que les résultats ne dépendent pas des conditions environnementales, ces expériences sont réalisées sur trois sites différents de la plateforme, sur différents canaux IEEE 802.15.4 sujets à divers niveau de bruit, pour plusieurs tailles de paquets.

Évaluation des interférences

Pour évaluer l'interférence possible entre les différentes technologies utilisant la bande ISM, les radios sont démarrées en mode RX sur les trois sites. Les capteurs échantillonnent la puissance radio mesurée par le matériel lorsque celui-ci n'est pas en train de recevoir une trame IEEE 802.15.4. Cette mesure donne la puissance radio brute reçue par l'antenne à considérer comme du bruit — nous l'appellerons donc le RNSI (*Received Noise Strength Indicator*) pour le distinguer

2. À Grenoble et à Lille, la plateforme se trouve dans le bâtiments de l'INRIA ; à Lyon, elle se trouve dans les locaux de l'INSA.

TABLE 4.1 – Caractéristique des sites & configuration des capteurs pour les expériences de caractérisation des liens sur la plateforme IoT-LAB.

Site	Capteurs en émission	Nombre moyen de voisins	Paquets par lien[a]	Taille des paquets[b]	Puissance d'émission
				octets	dBm
Lyon	18	5,0	90	33	−12
				84	
				126	
Grenoble	30	3,0	90	84	0
Lille	30	7,4	90	84	0

[a] C.-à-d. par émetteur & par canal.
[b] PDU de la couche liaison de données.

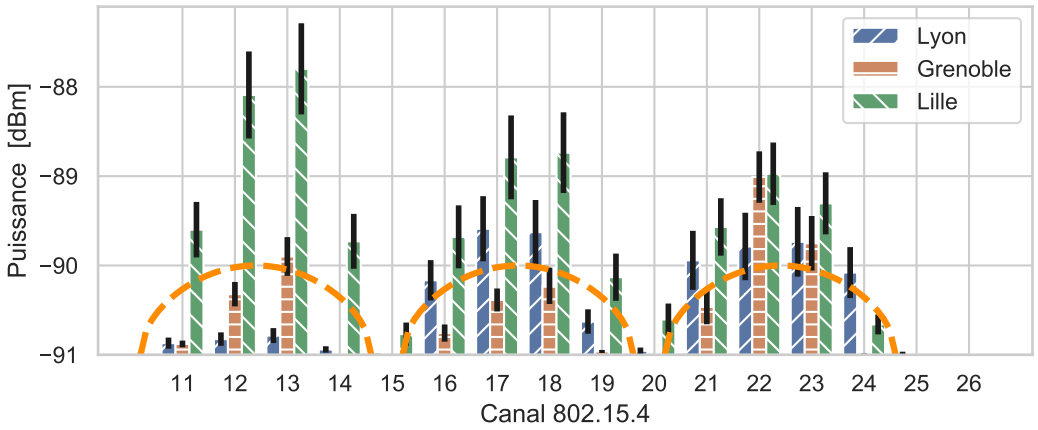


FIGURE 4.1 – Bruit moyen mesuré par des capteurs IEEE 802.15.4 de la plateforme IoT-LAB à Grenoble, Lille & Lyon, avec les intervalles de confiance sur la mesure. Les courbes en pointillés représentent les canaux IEEE 802.11 les plus utilisés (1, 6 & 11). Les canaux IEEE 802.15.4 numéro 15, 20, 25, & 26 offrent une moyenne très proche de −91 dBm et un écart-type quasi nul, aussi sont-ils peu visibles sur le graphe.

de la puissance radio d'une trame. Selon la norme IEEE 802.15.4 [10], « la valeur de l'ED[3] est une estimation de la puissance du signal reçu dans la largeur de bande du canal. Aucune tentative n'est faite pour identifier ou décoder les signaux sur le canal. La durée de mesure de l'ED, sur laquelle [est calculée] la moyenne, est égal à 8 périodes symbole », c'est-à-dire 128 µs. Cette mesure est utilisée entre autres pour mesurer l'occupation du canal lors d'un CCA.

Les résultats sont présentés dans la figure 4.1. Le bruit dépend clairement du canal et du site considéré. La corrélation avec les canaux de IEEE 802.11b/g/n y est aussi très visible : jusqu'à pratiquement +3 dB de puissance supplémentaire sur les canaux 12 et 13 à Lille où les deux technologies se chevauchent, par rapport au canal 15 sur ce même site.

L'impact de IEEE 802.11 sur IEEE 802.15.4 a déjà été étudié. SHIN, PARK et KWON [77] ont proposé un modèle analytique (appuyé sur des simulations) pour évaluer l'effet des interférences mutuelles entre ces deux technologies. Ils ont montré que l'interférence due au signal IEEE 802.11 devient négligeable sur le PER de IEEE 802.15.4 si le rapport des distances entre les émetteurs

3. La norme IEEE 802.15.4 utilise l'acronyme ED (*Energy Detection*) pour désigner cette mesure. Ici, nous préférons l'acronyme RSSI parce qu'il permet de distinguer les cas où nous parlons de la puissance radio perçue lors de la réception d'un paquet (*Received Signal Strength Indicator*, RSSI) de celle perçue lorsqu'aucune transmission n'est en cours (*Received Noise Strength Indicator*, RNSI).

et l'antenne réceptrice est supérieur à 8 (dans leur expérience, l'émetteur IEEE 802.11 à 8 m et l'émetteur IEEE 802.15.4 à 1 m). MARINA et al. [76] ont étudié expérimentalement le PDR (*Packet Delivery Ratio*) obtenu pour différentes longueurs de paquets IEEE 802.15.4 en présence de trafic IEEE 802.11g/n. Ils ont montré que le canal et la modulation de IEEE 802.11 ont tous deux une importance. ANGRISANI et al. [79] ont également étudié cette question sur un vrai banc d'essai et ont trouvé une corrélation entre le PER et le SNR; ils n'ont comparé leurs résultats à aucune formule théorique. BROWN et al. [100] ont proposé un modèle statistique pour prédire les pertes sur la base de l'observation du brouillage. Leur modèle est plus simple que le nôtre : il considère l'interférence comme un processus ON/OFF, qui détruit une transmission dès qu'il y a chevauchement. En fait, l'impact de l'interférence peut être moins important lorsque l'interférence n'est pas trop puissante; une étude plus fine de l'interférence est nécessaire.

En fait, des calculs simples suffisent pour appréhender l'impact de IEEE 802.11 sur IEEE 802.15.4 et ses éléments clefs. Le tableau 4.2 présente quelques caractéristiques d'émission et de réception des deux technologies. Si l'on considère que l'atténuation des deux signaux est la même (c'est-à-dire *grosso modo* que la distance entre les émetteurs et le récepteur est identique), le signal IEEE 802.15.4 aurait alors un SIR de −10 dB, ce qui rend la communication impossible même avec un facteur d'étalement de spectre effectif de 8 (+9 dB). Comme les réseaux IEEE 802.11 sont moins denses, ou, exprimé différemment, ont un bilan de liens plus élevé [113] de 6 à 10 dB, il est équitable de considérer le cas où l'émetteur IEEE 802.11 est plus éloigné que l'émetteur IEEE 802.15.4 ou derrière un mur ou une porte, ce qui peut atténuer la puissance du signal de 6 à 15 dB [136]. Cette atténuation diminue d'autant le niveau d'interférence et rend la communication IEEE 802.15.4 possible.

L'impact de la transmission IEEE 802.11 n'est ainsi pas négligeable et cette interférence doit être prise en compte, même si elle ne détruit pas toujours toutes les communications IEEE 802.15.4. Les méthodes d'échantillonnage du bruit utilisées dans la section 4.3 le prennent entre autres en compte, ainsi que tous le reste du bruit, quelque soit sa source.

Il est à noter qu'à l'inverse, IEEE 802.11 est peu perturbé par les interférences en provenance de IEEE 802.15.4, à cause de la différence de puissance de transmission et aussi de la largeur de canal utilisé :

> Pour un récepteur IEEE 802.11b, [IEEE 802.15.4] ressemble à un brouilleur à bande étroite, et le gain de traitement résultant des techniques d'étalement du spectre dans [IEEE 802.11b] permettra de réduire l'impact du brouilleur IEEE 802.15.4.
>
> — Norme IEEE 802.15.4, Annexe E

TABLE 4.2 – Comparaison de quelques caractéristiques de transmission entre IEEE 802.15.4 et IEEE 802.11.

	Largeur de canal	Puissance de transmission	SNR	
	MHz	dBm	même atténuation pour les deux signaux	−6 à −15 dB d'att. supplémentaire pour IEEE 802.11
IEEE 802.15.4[a]	2	0		
IEEE 802.11n	20	~20		
Ratio	−10 dB	−20 dB	−10 dB	−4 à +5 dB

[a] Valeurs données pour la couche physique utilisée pour les expériences : O-QPSK dans la bande des 2450 MHz.

4.1 Estimation du taux de perte à partir du RSSI

L'un des moyens proposés par la norme IEEE 802.15.4 pour évaluer la qualité de réception des paquets mesurée à la couche physique est le RSSI. Il mesure la puissance moyenne du signal reçu sur 8 symboles IEEE 802.15.4.

La tendance du RSSI en fonction de la distance dans la figure 4.2 corrobore les mesures déjà rapportées [72, 80]. Dans la figure, chaque point représente un lien, et sa position est donnée par la moyenne d'environ 2000 mesures, sur les 16 canaux fréquentiels disponibles. Deux groupes de points peuvent y être observés. À gauche, du côté des liens proches (en dessous de 6 m) le RSSI reste bien au-dessus de la sensibilité du récepteur (−91 dBm pour nos puces radio) et il suit une tendance clairement linéaire. Il reflète une relation linéaire entre la puissance reçue et $\frac{1}{d^\gamma}$, avec une bonne correspondance entre nos mesures et le modèle de sol à deux rayons ($\gamma = 4.12$). Il y a beaucoup de variabilité, car le coefficient de corrélation est seulement $r^2 = 0.652$. Cet exposant de distance se situe bien à l'intérieur du facteur d'affaiblissement $\gamma \in [1.90, 4.75]$ rapporté par HARA et al. [72].

Deuxièmement, pour les distances où le RSSI est souvent inférieur à la sensibilité (au-delà de 6 m, dans cette configuration), les mesures sont biaisées par les pertes de paquets. Elles ne sont donc pas utilisées pour l'ajustement linéaire, bien que les points restants suivent toujours la même tendance.

Le RSSI est donc très grossièrement corrélé avec la distance ; toutefois, la variabilité de cette corrélation montre que des liens longs peuvent offrir un très bon RSSI tandis que des liens relativement courts offrent un faible RSSI.

L'examen de la relation entre RSSI et le PER dans la figure 4.3 révèle qu'environ 95 % des liens ayant un RSSI moyen de −85 dBm (c'est-à-dire 6 dB au-dessus du seuil de puissance de réception radio) présentent un PER de moins de 20 %. Ils peuvent être considérés comme *bons*, et être utilisés sans problème. Cependant, le RSSI donne peu d'informations pour les liens dont le RSSI moyen est plus faible : la grande majorité des liens sont encore utilisables même lorsque RSSI est proche du seuil de puissance de réception radio (c'est-à-dire −91 dBm), alors que certains autres sont à éviter car ils perdent 90 % des paquets ou davantage.

Dans un environnement extérieur, également avec des radios IEEE 802.15.4, ZENNARO, NTAREME et BAGULA [80] ainsi que d'autres après eux [91] ont trouvé le même type de valeur seuil RSSI en dessous de laquelle la qualité du lien n'est pas du tout garantie, bien que certains liens puissent être bons.

Ces résultats ne sont pas si surprenants. Le RSSI indique la puissance du signal reçue dans la bande fréquentielle correspondant au canal IEEE 802.15.4 durant la réception de l'en-tête de trame. Cependant, il ne fournit pas d'informations sur l'intensité du bruit ni sur le brouillage. C'est pourquoi IEEE 802.15.4 propose une autre métrique appelée LQI, qui est une estimation de la qualité de réception d'un paquet, pas seulement de sa puissance.

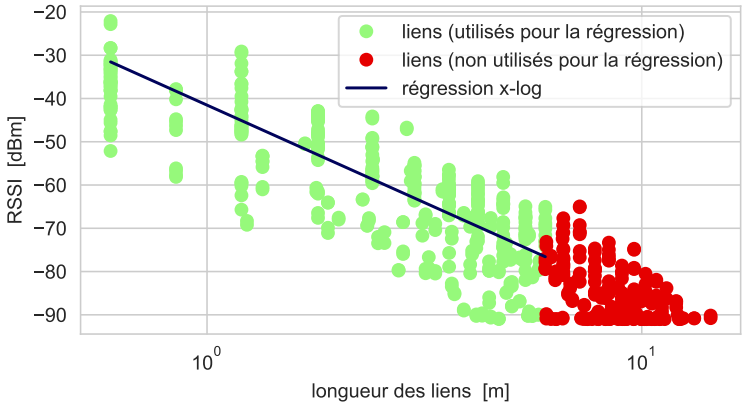


FIGURE 4.2 – Relation entre le RSSI moyen d'un lien et sa longueur (distance entre l'émetteur et le récepteur). Chaque point est une liaison entre deux des 28 capteurs du banc d'essai. Lorsque le hardware radio annonce une puissance ≤ −91 dBm (sensibilité minimale de la radio), des paquets commencent à manquer et faussent l'ensemble des valeurs ; la régression linéaire est donc calculée uniquement en dessous de 6 m.

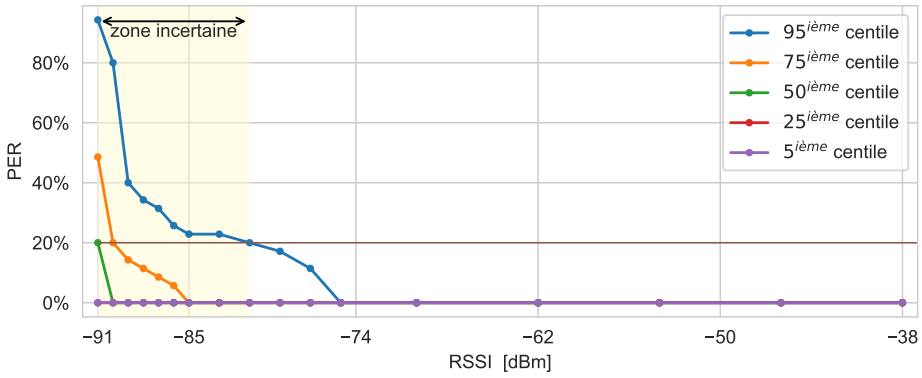


FIGURE 4.3 – Le PER d'un lien en fonction de la moyenne du RSSI perçu. Exemple d'interprétation : pour les liens avec un RSSI moyen de 81 dBm, 95 % d'entre eux présentent un PER en-dessous de 20 %.

4.2 Estimation du taux de perte à partir du LQI

Le RSSI ne fournissant pas d'informations sur l'intensité du bruit ou des interférences qu'a rencontré la trame, IEEE 802.15.4 propose une autre métrique appelée LQI (*Link Quality Indicator*) :

> La valeur du LQI est une caractérisation de la puissance (*strength*) et/ou de la qualité de réception d'un paquet. La mesure peut être effectuée au moyen de l'ED [4], d'une estimation du rapport signal à bruit ou d'une combinaison de ces méthodes. [...] Les valeurs minimum et maximum du LQI (`0x00` et `0xff`) doivent être associées aux signaux conformes de qualité la plus basse et la plus élevée détectables par le récepteur [...]
>
> — Norme IEEE 802.15.4 [10]

La norme ne définit donc pas de façon précise comment calculer le LQI, et se contente de parler de *puissance* et de *qualité*. Sa valeur dépend beaucoup de l'implémentation de cette mesure. Elle est parfois lié au PER (comme p. ex. le dit PAPADOPOULOS et al. [114]), mais cela n'est pas clairement statué par la norme. Les émetteurs-récepteurs radio CC2420, CC2520 et CC2538, à l'instar de la norme, laissent entièrement le choix du calcul du LQI au logiciel. Ils fournissent une mesure de corrélation basée sur quelques symboles de la trame, et suggèrent l'utilisation d'une « combinaison du RSSI et des valeurs de corrélation » [117, 119, 120]. La fiche technique de l'émetteur-récepteur AT86RF231 utilisé par les capteurs M3 du banc d'essai, quant à elle, explique clairement comment la puce calcule le LQI, et indique explicitement un lien avec le PER :

> La valeur du LQI de l'AT86RF231 est effectuée par une mesure de la qualité de la liaison qui peut être décrite avec le taux d'erreur des paquets (PER) pour ce lien. [...] Le [récepteur] radio utilise les résultats de corrélation de plusieurs symboles dans une trame pour déterminer la valeur du LQI. Par exemple, [la figure 4.4] montre le [taux] d'erreur de paquet conditionnel lors de la réception d'une certaine valeur de LQI.
>
> — Fiche technique de l'AT86RF231 [118]

L'étude présentée ici mesure expérimentalement la relation entre le LQI et le PER, en utilisant la technique de corrélation implémentée par les puces AT86RF231. En première approche, dans la section 4.2.1, l'étude est restreinte au site d'IoT-LAB de Grenoble. Elle y montre qu'il y est possible d'utiliser cette métrique pour identifier les liens de bonne qualité, même lorsque la puissance du signal sur ce lien est très faible. Dans la section 4.2.2, en étudiant les limites de cette approche, l'étude est étendue à l'un des deux autres sites d'IoT-LAB décrits dans l'introduction — le site de Lille — afin de présenter des résultats plus généraux.

4.2.1 Critère de classification des liens

La section 4.1 et la figure 4.3 ont montré que le RSSI n'est pas suffisant pour classer les liens lorsque la puissance du signal est inférieure à environ −85 dBm. Quand le RSSI n'aide plus, le LQI commence à fournir des informations intéressantes pour différencier les bons des mauvais liens. La figure 4.5 résume les informations fournies par le LQI en-dessous de −85 dBm. Elle montre que, même lorsque le RSSI atteint le seuil de réception radio (−91 dBm), le LQI permet toujours de considérer 40 % des liens comme des liens *bons*, d'après les critères définis par la suite. Même la métrique ETX, pour ces liens marginaux, est souvent une métrique très instable [87].

La figure 4.6 présente la corrélation entre le PER d'un lien et le LQI moyen perçu sur ce lien [5]. Trois zones sont identifiables, et sont également résumées dans le tableau 4.3. Du côté droit, les

4. Cf. note 3 en page 76 au sujet de l'utilisation de l'acronyme ED dans la norme.

5. Il faut noter que les valeurs basses de LQI sont sous-représentées car elles impliquent bien souvent un paquet perdu. Dans la figure 4.6, il a été très difficile d'observer des paquets de LQI inférieur à 100.

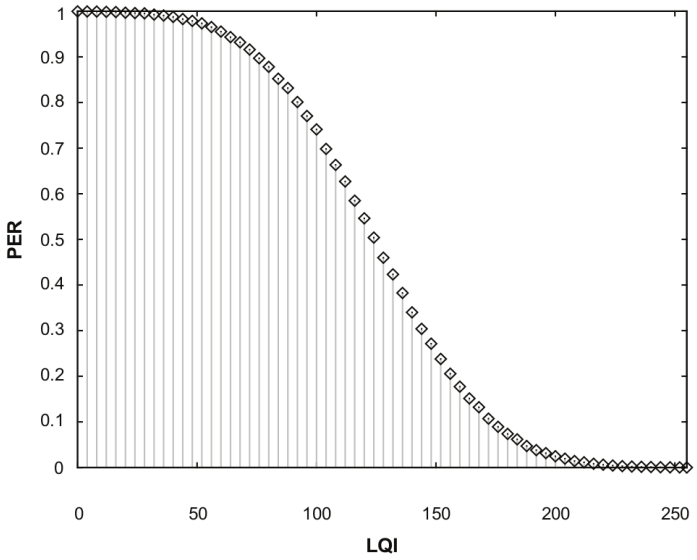
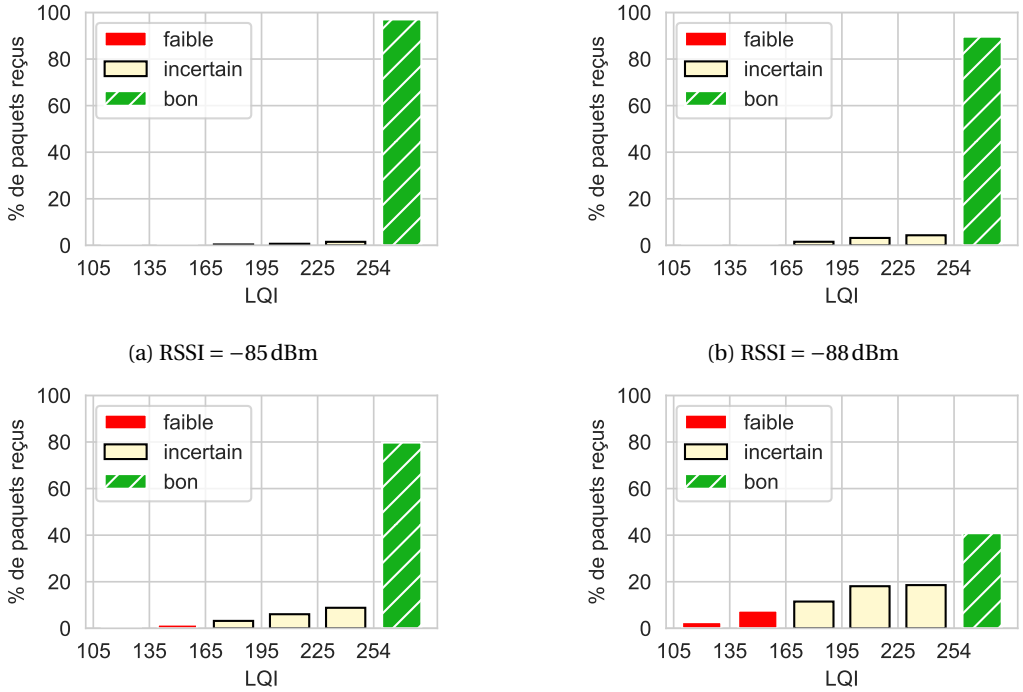


FIGURE 4.4 – « PER conditionnel vs. LQI. Les valeurs sont prises à partir de trames reçues d'une longueur PSDU (*Physical Service Data Unit*) de 20 octets sur des canaux de transmission avec des écarts raisonnables de délai de propagation par trajets multiples faibles. [...] Le PER indiqué [ici] est basé sur un nombre considérable de transactions. » (extrait de la fiche technique de l'AT86RF231 [118])



(a) RSSI = −85 dBm

(b) RSSI = −88 dBm

(c) RSSI = −90 dBm

(d) RSSI = −91 dBm

FIGURE 4.5 – Distribution du LQI pour une trame reçue avec une certaine puissance (RSSI). La classification suit les critères donnés dans le tableau 4.3. La barre la plus à droite est pour $LQI = 255$ seulement, les autres barres agrègent 30 valeurs consécutives.

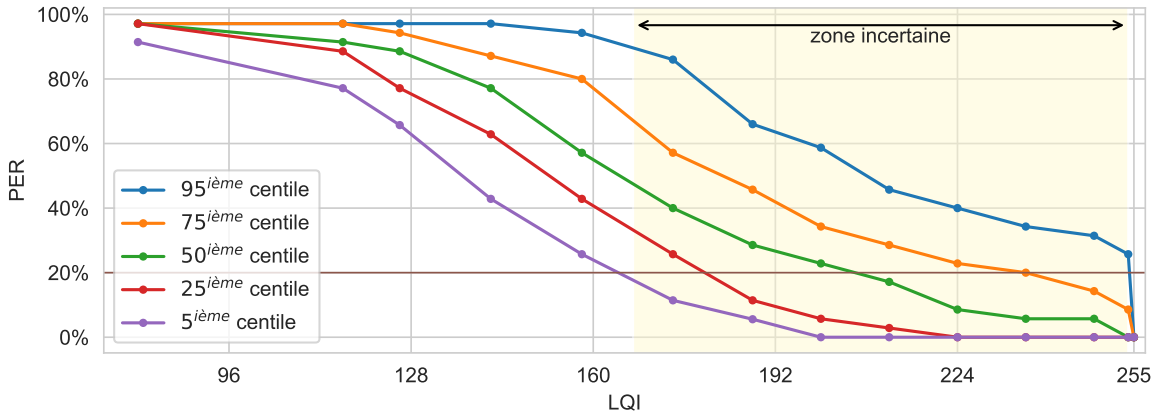


FIGURE 4.6 – Distribution du PER d'un lien en fonction de la moyenne du LQI observé sur ce lien. La zone incertaine correspond aux valeurs où le LQI n'est pas suffisant pour classer les liens comme *bons* ou *faibles*. Notez bien que le point LQI = 255 n'est pas inclus dans la zone incertaine.

TABLE 4.3 – Classification des liens d'après le LQI mesuré. L'équivalent en matière de PER est donné pour 95 % des cas.

Appellation	Critère	Équivalent en matière de PER	Recommandation
bon	LQI = 255	PER < 20%	Un lien qui peut être utilisé sans crainte
incertain	165 ≤ LQI < 255	indécidable	Trop peu d'informations sont nécessaires pour classifier ce lien; il faut préférer des liens connus et *bons*, ou préciser les mesures au sujet de ce lien pour le classifier
faible	$LQI \leq 165$	PER > 20%	Très certainement un lien trop faible : à annoncer comme tel au protocole de routage

liens qui exposent un LQI = 255 peuvent être considérés comme *bons*, c'est-à-dire que leur PER < 20% pour 95 % des cas; cependant, dès que LQI tombe en dessous de 254, la quantité de liens *faibles* (c'est-à-dire de PER > 20%) ne sera plus négligeable. Plus loin à gauche, où le LQI ≤ 165, la plupart des liens (toujours 95 %) sont *faibles*. Entre ces deux limites, les liens ne peuvent pas être immédiatement classés comme *bons* ou *faibles*. Pour les différencier, le capteur doit y consacrer plus de ressources. Dans la suite, ces liens seront appelés *liens incertains*.

On note la forme ressemblante de cette courbe avec la courbe théorique donné par la fiche technique de l'AT86RF231, présentée en figure 4.4. Les valeurs de PER ne sont toutefois pas exactes pour un LQI donné comme aurait tendance à le sous-entendre la fiche technique, mais sujets à la variabilité de l'environnement. De plus, le signal de qualité la plus basse détectable par le récepteur semble plutôt engendrer un LQI un peu en-dessous de 100 plutôt que d'atteindre 0. Cela montre encore la difficulté d'interprétation de la valeur précise du LQI.

Ce groupe de liens *incertains* est similaire à celui défini par WOO et CULLER [64] ainsi que par d'autres [65, 68, 78]. Moins de liens se trouvent cependant dans cette portion lorsque le LQI est utilisé au lieu de RSSI, comme illustré dans la figure 4.5. Le LQI est donc une métrique capable de prédire assez précisément le PER des liens.

TABLE 4.4 – Distribution de la qualité du lien dans le sens de l'émission ($X \rightarrow Y$) en fonction de la qualité du lien dans le sens de la réception ($Y \rightarrow X$).

		$(X \rightarrow Y \mid Y \rightarrow X)$ [%]		
		bon	incertain	faible
$(Y \rightarrow X)$	bon	**95,6**	4,4	0,0
	incertain	34,6	**58,6**	6,8
	faible	1,4	44,1	**54,5**

4.2.1.1 Qualité des liens en sens inverse

En général, les modèles de propagation simples supposent que les liens entre les capteurs sont symétriques, c'est-à-dire qu'ils ont le même PER dans les deux sens. Cette hypothèse est valable pour des liens forts qui offrent un RSSI élevé et pour lesquels une petite différence dans les circuits radio ou un brouillage localisé n'ont pas d'impact sur le PER. Cette hypothèse peut être remise en cause lorsque le RSSI approche le seuil de puissance de réception radio et que la qualité des liens varie fortement pour une puissance de réception donnée [114].

Pour évaluer la prévalence des liens asymétriques, le tableau 4.4 rapporte la proportion de liens dans une classe donnée pour le sens de l'émission en fonction de la classe dans le sens de la réception. Les données confirment que le cas symétrique est le plus fréquent. Les liens *incertains* exigent peut-être de consacrer plus d'efforts pour savoir s'il faut les utiliser ou non : plus d'un tiers est en fait bon dans l'autre sens, ce qui suggère que le PER pourrait être acceptable.

4.2.2 Limites de l'approche et de l'estimateur

Le LQI (dans l'état de l'implémentation de l'AT86RF231) est capable de **capturer la qualité d'un signal**, quelle que soit sa puissance. Il permet ainsi de distinguer les liens qui sont sujets à plus ou moins de perturbations. Les résultats obtenus montrent qu'il est possible d'aller plus loin qu'avec le RSSI, en classant les liens en deux catégories : les liens *bons* et les liens *faibles*, même lorsque le RSSI observé sur le lien est faible. Cette classification rappelle la métrique simple *hop count with weak links* des anciennes versions de LOADng [47], qui permet d'éviter les routes comportant des liens faibles tant que d'autres solutions existent. VAN TRIMPONT et al. [107] l'ont expérimenté sur le réseau G3-PLC (*G3-Power Line Communication*) [139, 129] (avec LOADng comme protocole de routage, conformément à la norme) en rapportant des résultats meilleurs ou similaires qu'avec la métrique *hop count* lorsque le nombre de voisins est suffisant. Une troisième catégorie existe, les liens *incertains*, où une étude encore plus poussée doit être menée.

La figure 4.7 étudie **la variabilité du LQI**, en présentant l'écart-type du LQI sur chaque lien. La figure ne présente que les liens qui présentent un faible RSSI ($\overline{\text{RSSI}} \leq -91\,\text{dBm}$), car pour des valeurs supérieures de RSSI, le LQI reste la plupart du temps à 255 comme présenté sur la figure 4.5. À droite, la très-faible variabilité montre que les liens *bons* restent vraiment stables, ce qui confirme que ces liens peuvent réellement être considérés comme utilisables, même si leur RSSI est faible. Cependant, lorsque le LQI diminue, la variabilité du LQI est loin d'être négligeable [6]. C'est une raison de plus de considérer un lien *incertain* comme suspect et de le vérifier soigneusement avant de l'utiliser, et à plus forte raison un lien *faible*.

Un autre point important qui peut être soulevé ici est que **la valeur du LQI n'est pas portable d'un capteur à l'autre** (hormis peut-être pour les cas 255 et 0, signifiant *le meilleur* / *le moins bon*

6. Cf. note 5 page 80. Dans la figure 4.7, en dessous de ≈ 222, le LQI semble garder la même variabilité toujours pour cette même raison de manque d'échantillons dans les valeurs basses du LQI.

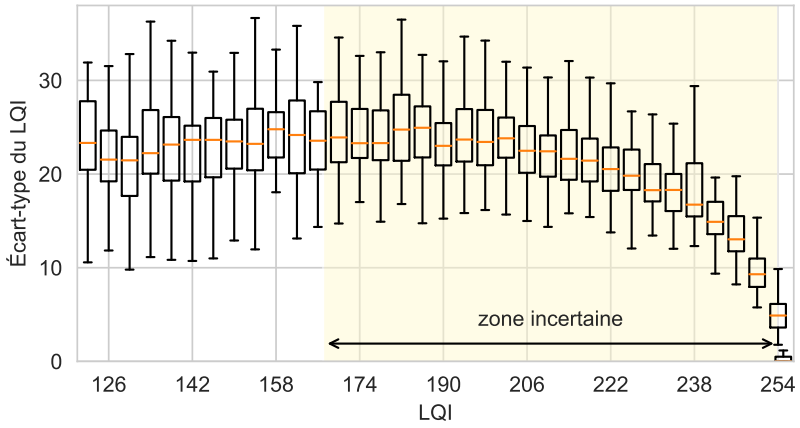


FIGURE 4.7 – Variabilité du LQI. Seuls les liens qui offrent un RSSI moyen ≤ −91 dBm sont pris en compte. L'écart type est calculé pour chaque lien séparément.

lien possible), puisque le calcul du LQI est dépendant de l'implémentation. L'étude présentée ici ne s'occupe que du cas des puces AT86RF231, les tests seraient à répéter pour une autre puce radio. Cela contraindrait le protocole de routage à utiliser une autre grandeur, comme une classification en deux ou trois classes de liens, plutôt qu'à transmettre directement la valeur du LQI d'un capteur à l'autre.

Enfin, un autre défaut du LQI est qu'une mesure ne représente pas un lien, mais un paquet. Le bruit rencontré lors de la transmission d'un paquet n'a pas forcément de raison d'être identique lors de la transmission d'une autre trame, particulièrement lorsque la source du bruit est une technologie où l'interférence avec IEEE 802.15.4 est ponctuelle et imprévisible, comme du trafic IEEE 802.11 ou des décharges des magnétrons des fours à micro-ondes. **La valeur moyenne du LQI**, utilisée dans cette section comme estimateur du LQI des paquets à venir, **peut ainsi devenir un très mauvais estimateur**. Sur d'autres sites d'IoT-LAB où le niveau de bruit est plus élevé (cf. figure 4.1 discutée plus tôt dans l'introduction), les résultats sont beaucoup moins précis. La figure 4.8 compare le site de Lille avec celui de Grenoble sur plusieurs canaux. Elle montre que la prédiction du PER est relativement précise pour le canal 26 par exemple (l'intervalle intercentiles est fin), mais elle montre aussi que sur les canaux 22 et 23, la précision est beaucoup moins bonne et même un LQI moyen de 255 ne garantit pas un PER de 0 %.

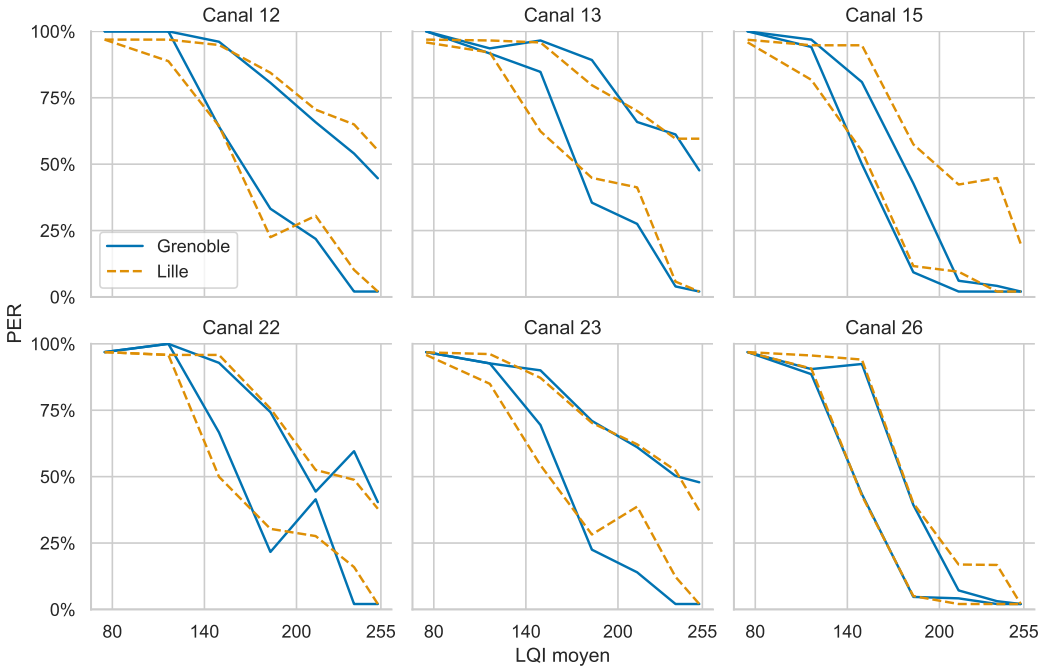


FIGURE 4.8 – Cinquième et quatre-vingt-quinzième centiles de la distribution du PER des liens en fonction de la moyenne du LQI observé sur ce lien.

4.3 Estimation du taux de perte à partir du niveau de bruit

Le LQI est donc un bon estimateur de la qualité des paquets reçus, mais il est difficile de prévoir sa valeur pour les paquets à venir. L'idée présentée dans cette section découpe la mesure de la qualité du lien en deux parties. D'un côté, une étude statistique du bruit observé sur le canal est faite afin de prévoir son impact sur la transmission avec plus de précision. De l'autre côté, on a une mesure matérielle de la puissance associée aux paquets en provenance d'un voisin donné. Ces deux informations conduisent ensuite naturellement au calcul du SNR, puis du PER du lien par applications de formules théoriques.

Cette approche a l'avantage de combiner l'approche des estimateurs statistiques et des estimateurs physiques. L'estimation du bruit est statistique et indépendamment des capteurs voisins ; elle est tenue à jour au cours de la vie du capteur sans surcharge de transmission radio et avec très peu d'exigences de performances. La mesure de la puissance des paquets transmis sur un lien est entièrement physique et peut être estimé à partir de très peu de paquets échangés sur le lien — dans beaucoup de cas, un seul paquet suffit pour avoir une précision satisfaisante. Ainsi, l'estimation du PER avec la méthode décrite ici combine efficacement rapidité et données statistiques.

La section 4.3.1 explique comment sont mesurés les deux paramètres d'entrée du modèle, le niveau de bruit et la puissance des liens. La section 4.3.2 présente comment le PER est calculé à partir de ces données. Enfin, la section 4.3.3 présente les résultats expérimentaux obtenus sur la plateforme IoT-LAB et la précision de l'estimation du PER de chaque lien.

4.3.1 Mesure des paramètres physiques radio

Il y a principalement deux entrées au modèle de calcul du PER décrit ici. Tout d'abord le niveau de bruit ; pour celui-ci, deux approches de mesure sont présentées, l'une par un échantillonnage continu du bruit afin d'être aussi précis que possible (section 4.3.1.1), l'autre par un échantillonnage cyclique du bruit, plus adaptée pour une implémentation sur capteurs (section 4.3.1.2). Ensuite, la puissance des paquets reçus sur un lien donné (section 4.3.1.3).

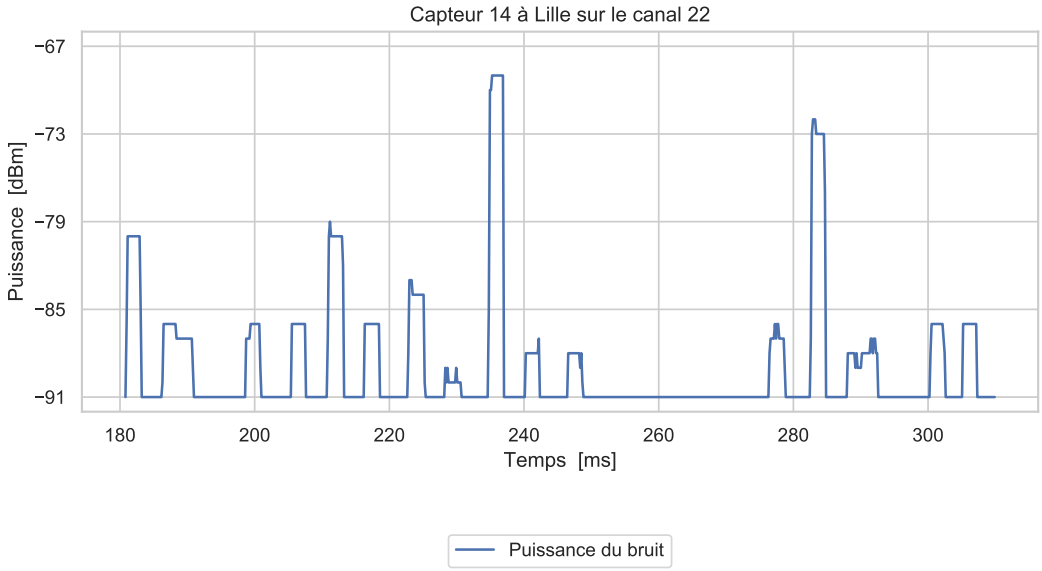


FIGURE 4.9 – Capture continue du niveau de bruit observé par la radio des capteurs durant les expérimentations d'IoT-LAB.

4.3.1.1 Mesure du niveau de bruit par échantillonnage continu

Pour obtenir une caractérisation très précise du bruit, les capteurs enregistrent en continu la valeur de l'ED [7], observé sur le canal radio. Ces données sont stockées dans la RAM du capteur et extraites dans un deuxième temps via le port série afin de ne pas perturber la mesure en faisant travailler le capteur sur la radio et le port série simultanément. Chaque point de mesure est une moyenne de la puissance reçue dans la bande de fréquences considérée pendant 128 microsecondes. L'enregistrement dure 1,38 seconde pour 8300 échantillons (période de mesure de 166 µs, un peu supérieur au temps de mesure à cause du temps d'aller-retour sur le bus SPI (*Serial Peripheral Interface*)).

La figure 4.9 présente un tel enregistrement. On y voit très clairement des salves courtes de bruit certainement dues aux émissions d'autres technologies, entre autres les points d'accès et les stations IEEE 802.11 autour du capteur.

Cette méthode de mesure du bruit n'est pas applicable pour un capteur de faible puissance dans des déploiements réels, et servira plutôt ici comme référence. En effet, la radio doit être allumée et occupée à la mesure pendant de longues périodes de temps ; cela impose aussi une lourde charge au processeur pour gérer les entrées et les sorties sur le bus SPI ; cela produit beaucoup de données et nécessite une certaine puissance de calcul pour les traiter. De plus, une salve de mesures de bruit relativement courte peut facilement ne pas être représentative des conditions à long terme du canal, et doit être mise à jour chaque fois que l'environnement est supposé avoir changé — potentiellement souvent.

4.3.1.2 Mesure du niveau de bruit par échantillonnage cyclique

Pour pouvoir intégrer l'estimation du PER dans un capteur, une deuxième méthode permet d'obtenir des informations sur le bruit en se basant sur une mesure cyclique du bruit. Le ratio d'activité cyclique est une technique courante pour réduire la consommation d'énergie au niveau de la couche liaison de données : la radio est allumée périodiquement, à des instants précis, lorsque le capteur s'attend à recevoir un paquet d'un de ses voisins. C'est le cas des protocoles MAC syn-

7. Cf. note 3 en page 76 au sujet de l'utilisation de l'acronyme ED dans la norme.

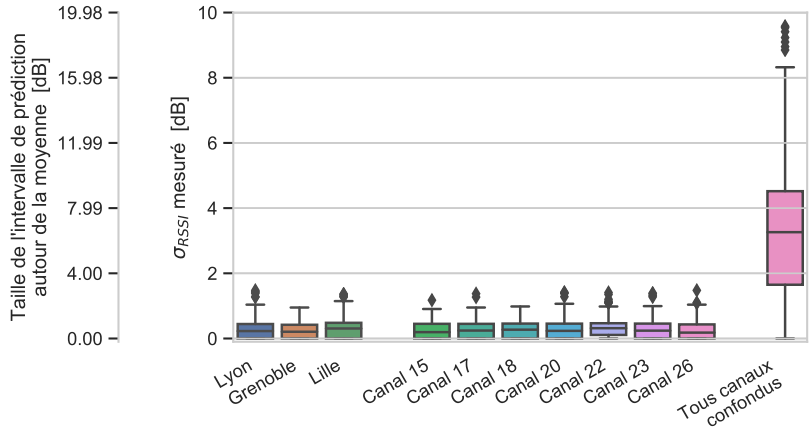


FIGURE 4.10 – Écart-type du RSSI et taille de l'intervalle de prédiction autour de sa moyenne pendant les expériences.

chrones (p. ex. les modes TSCH et *beacon-enabled* de IEEE 802.15.4 [10]) comme asynchrones (p. ex. ContikiMAC [90]). Si un SHR (*Synchronization Header*) est reçu, la radio reste en mode RX pour recevoir la trame complète. Si le capteur ne détecte pas de SHR, alors il peut mesurer la puissance du bruit ambiant avant d'éteindre la radio. Ainsi, le capteur est capable d'échantillonner le RNSI quasiment sans consommation d'énergie supplémentaire. Pour représenter ces mesures de façon succincte et aussi exploitable, elles sont stockées sous forme d'histogramme de distribution du RNSI, facilement stockable dans la mémoire du capteur. Cette distribution peut être affinée et maintenue à jour au fur et à mesure que des mesures sont remontées par le ratio d'activité cyclique pour refléter l'état courant de l'environnement, en pondérant les nouveaux échantillons plus que les anciens, par exemple avec une moyenne glissante exponentielle ou pondérée.

4.3.1.3 Estimation de la puissance du signal sur un lien

En plus du bruit du canal, le modèle d'estimation du PER exige de connaître la puissance des paquets sur un lien. La valeur pour un paquet déjà reçu est donnée par le matériel, par la valeur du RSSI jointe au paquet. La figure 4.10 présente l'écart-type de la puissance des paquets reçus sur un lien [8], telle que mesurée expérimentalement sur la plateforme IoT-LAB. En plus de l'écart-type, la figure présente l'intervalle de prédiction de cette puissance (c.-à-d. l'intervalle qui comprendra la puissance du prochain paquet, avec $\alpha = 95\,\%$ de confiance). L'intervalle de prédiction est calculé pour une variable de loi normale de moyenne et d'écart-type inconnus à l'aide de la formule 4.1, avec $t_{\frac{\alpha}{2},n-1}$ le quantile d'ordre $1-\alpha/2$ de la loi de Student à $n-1$ degrés de liberté et $n = 90$ le nombre de paquets envoyés sur chaque lien.

$$I_{\text{prédiction}} = \left[\mu_{\text{RSSI}} \pm t_{\frac{\alpha}{2},n-1} \times \sigma_{\text{RSSI}} \times \sqrt{1+\frac{1}{n}} \right] \tag{4.1}$$

En pratique, comme le montre la figure, sur un même canal, l'écart-type du RSSI, est globalement inférieur à 1 dB sur un canal donné, ce qui donne une prédiction de la valeur du RSSI à ±2 dB. Comparé à la précision requise par l'estimation du PER (que nous verrons dans les résultats en section 4.3.3, et surtout dans la figure 4.13), il suffit en première approche d'utiliser le RSSI d'un seul paquet pour estimer le RSSI attendu des autres paquets de ce lien. L'estimation pourra

8. C'est-à-dire que l'écart-type est calculé sur les mesures de puissances groupées par émetteur, par récepteur, et (hormis pour la colonne de droite) par canal.

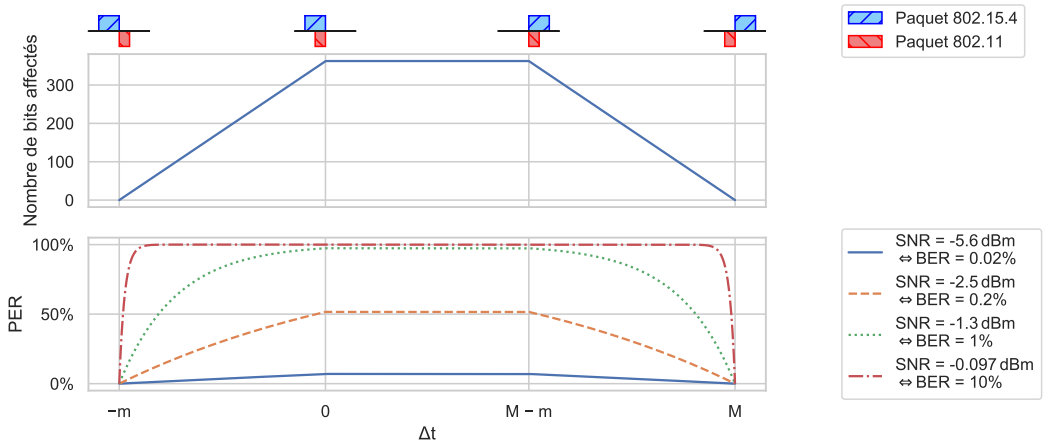


FIGURE 4.11 – Interférence entre une trame IEEE 802.15.4 de durée M et une interférence de durée m, pour une puissance impliquant un BER donné. Ici, $M = 4{,}10\,\text{ms}$ et $m = 1{,}45\,\text{ms}$, ce sont des valeurs représentatives par rapport aux expériences. Δt est la différence (dans le temps) entre l'instant de début d'émission des deux paquets.

toujours être affinée par la réception de quelques paquets supplémentaires, si le lien paraît utilisable.

La dernière colonne de la figure 4.10 montre que l'écart-type du RSSI est bien plus grand si on le mesure tous canaux confondus. Cette colonne montre que l'étude d'un canal ne suffit pas pour estimer le RSSI des autres canaux.

Les mesures de la stabilité du RSSI ont été réalisées principalement dans un environnement statique. Le mouvement peut causer des affaiblissements du gain du canal, et perturber les résultats prévus, comme cela a été observé dans des bureaux ou des parkings fréquentés [115] ; toutefois, les fluctuations du RSSI seront perçues par le capteur récepteur au cours de l'utilisation du lien, qui pourra ajuster l'estimation du RSSI du lien.

4.3.2 Modèle de calcul du PER

4.3.2.1 Approche théorique

Le signal radio physique envoyé par un capteur à un autre offre un certain SNR, que les mesures de la section précédente permettent d'estimer. À partir de ce SNR, le BER est calculé en utilisant la formule présente dans la norme IEEE 802.15.4 [8, version de 2006], rappelée dans la formule 4.2. Le PER est ensuite dérivé à partir du BER et de la longueur du paquet (formule 4.3).

$$\text{BER} = \frac{8}{15} \times \frac{1}{16} \times \sum_{k=2}^{16} -1^{k} \binom{16}{k} e^{\left(20 \times \text{SNR} \times \left(\frac{1}{k} - 1\right)\right)} \tag{4.2}$$

$$\text{PER} = 1 - \left(1 - \text{BER}\right)^{\text{longueur du paquet}} \tag{4.3}$$

Il a été montré que la longueur de la trame joue un rôle important dans l'estimation du PER, particulièrement pour les liens dont le PER est bas [116]. Le modèle présenté ici prend en compte la taille de la trame. J'ai montré dans la figure 4.9 que le bruit était principalement présent sous forme de salves courtes de puissance donnée, certainement dues à des interférences avec une autre technologie, comme des trames IEEE 802.11 ou encore des émissions sporadiques de four à micro-ondes. Or l'interférence peut avoir un impact sur une trame IEEE 802.15.4 même si elles ne se chevauchent que de quelques symboles : l'interférence pourrait ne masquer qu'un seul symbole, mais il en résulterait toujours un FCS erroné, de sorte que la trame entière serait perdue.

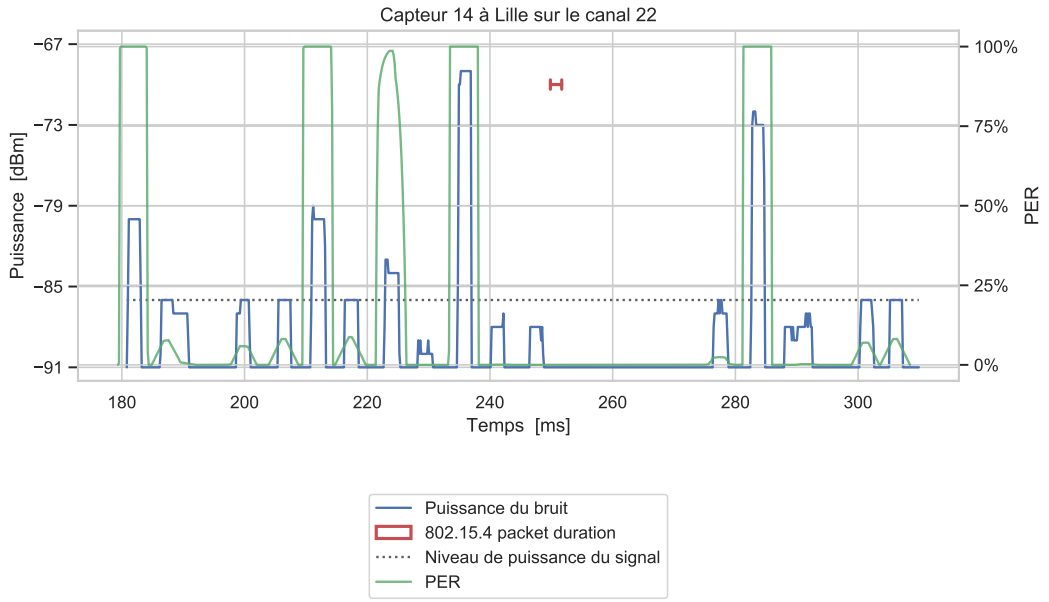


FIGURE 4.12 – Capture continue du niveau de bruit (cf. figure 4.9), ainsi que la probabilité de perte d'un paquet de taille 84 octets (= 180 symboles O-QPSK en comptant l'en-tête physique = 2,88 µs) reçu avec ce SNR, pour une puissance de −86 dBm.

Comme le montre la figure 4.11, l'intervalle de temps pendant lequel il y a collision entre une trame IEEE 802.15.4 et une interférence d'une durée m donnée (p. ex. une trame IEEE 802.11) est ainsi plus grand que la durée de l'interférence elle-même — sa durée est précisément la somme des durées des deux transmissions, et la zone d'influence de l'interférence est ainsi multipliée par un facteur pouvant aller jusqu'à $(M+m)/m$ dans le cas d'une interférence importante. Le modèle présenté ici prend en compte cette remarque.

4.3.2.2 Calcul du PER à partir des mesures continues du bruit

La mesure continue fournit une description très précise au cours du temps du niveau de bruit auquel un paquet doit faire face : pour une puissance de signal donnée, le SNR résultant peut être calculé pour n'importe quel instant le long de la salve de mesure — et donc la probabilité de perte d'un paquet de puissance donnée en appliquant les formules théoriques ci-dessus. C'est de cette façon qu'est calculée la courbe représentant la probabilité de perte d'un paquet sur la figure 4.12. On y voit que la démodulation du paquet commence à être perturbée lorsque le niveau de bruit atteint la puissance du signal [9], et la démodulation devient quasiment impossible lorsque le bruit est deux fois plus puissant que le signal (+3 dB). Comme le paquet échangé sur le lien peut arriver à tout instant, le PER global du lien est la moyenne pour toutes les positions possibles du paquet le long de l'enregistrement (l'intégrale sur la figure 4.12 de la courbe du PER).

Cette mesure nous donne une bonne illustration de ce qui se passe exactement lors de la transmission d'un paquet. Toutefois, elle ne peut pas être implémentée dans les conditions réelles, à cause de ses exigences en termes de complexité et de sollicitation matérielles. Pour une implémentation réelle, je propose de mesurer le bruit de façon cyclique.

9. À cause de l'étalement de spectre utilisé par la modulation O-QPSK de IEEE 802.15.4, la communication est pourtant toujours possible lorsque le SNR est à 1.

4.3.2.3 Calcul du PER à partir des mesures cycliques du bruit

La mesure cyclique du bruit produit un histogramme représentant la distribution du niveau de bruit sur le canal, chaque valeur représentant la probabilité d'apparition d'un niveau de bruit. À partir de cet histogramme, je dérive la distribution du SNR, pour enfin calculer le PER. La longueur des trames envoyées joue sur la probabilité de réception, comme décrit en section 4.3.2.1 et en figure 4.11, à cause du phénomène d'agrandissement de la zone d'influence d'une interférence. Cet élément est pris en compte en modifiant l'histogramme de la distribution du bruit. Chaque valeur de probabilité où le bruit n'est pas nul est multipliée par un facteur $(M+m)/m$ (c.-à-d. toutes les valeurs associées à un bruit supérieur à −91 dBm, valeur minimale que la radio AT86RF231 peut fournir et représentant une absence de bruit).

Une hypothèse majeure de cette méthode est la durée typique de l'interférence, c'est-à-dire la valeur de m, difficilement mesurable lors de la mesure cyclique du bruit. Une mesure expérimentale à l'aide des données remontées par les capteurs lors de la mesure continue du niveau de bruit a conduit à adopter la valeur de 1,45 milliseconde, qui correspond par ailleurs à une balise IEEE 802.11 de taille 180 octets envoyée à 1 Mbit/s. La durée typique des perturbations dépend entre autres de l'heure dans la journée : elle est plus courte durant la journée, parce que les paquets de données, bien que potentiellement plus longs, sont envoyés à un débit plus élevé. Pour comparaison, les trames IEEE 802.15.4 envoyées pendant nos expériences durent environ 1, 2,5 et 4 millisecondes (pour un taille de PDU en couche liaison de données de 33, 84 et 126 octets respectivement).

4.3.3 Expérimentations sur IoT-LAB

Les expériences décrites ci-dessous valident par la pratique le modèle théorique présenté dans la section précédente. La section 4.3.3.3 donne des indications quant à la précision de l'estimation du PER.

4.3.3.1 Mesure expérimentale du PER vrai

Afin de pouvoir comparer les résultats du modèle à des valeurs expérimentales, le PER vrai des liens est calculé en envoyant des dizaines de milliers de paquets sur la plateforme (90 paquets par émetteur et par canal). Lors de ces expériences de mesure du PER, les capteurs mesurent aussi le niveau de bruit avec les deux méthodes décrites en sections 4.3.1.1 et 4.3.1.2.

Comme attendu, beaucoup de liens sont très bons, et offrent un PER de 0 % (aucune perte), et quelques-uns, très mauvais, offrent un PER proche de 100 %. L'intervalle de confiance du PER est donc calculé avec le score de Wilson [50], plus à même de gérer ces cas extrêmes que l'approximation normale usuelle.

4.3.3.2 Visualisation des résultats

La figure 4.13 présente quelques-uns des résultats des expériences. Chaque graphique correspond à un capteur récepteur sur un canal. Sur chacun sont représentés les liens détectés vers d'autres capteurs d'après leur PER vrai et leur puissance, l'estimation du PER obtenue avec les deux méthodes du modèle, et la distribution du RNSI — histogramme et CCDF (complémentaire de la fonction de répartition), cette dernière indiquant la probabilité que le RNSI soit supérieur à une valeur de signal donnée. Comme attendu, le PER vrai des liens est lié avec le niveau de bruit : il reste à 0 % sur les canaux sans interférences (canal 26 p. ex.) même pour des petites puissances, alors que sur les canaux chargés, il augmente avec le niveau de bruit. Le modèle dans l'ensemble capture très bien ce détail, et produit une estimation proche de ces points de référence.

Sur certains graphiques, l'estimation obtenue avec la mesure continue du bruit diffère de l'estimation obtenue avec la mesure cyclique du bruit (p. ex. pour le capteur 94 à Grenoble, pour les valeurs de faible puissance). Il y a deux explications à cela. Premièrement, l'estimation par mesure continue du bruit utilise une salve de mesures de bruit qui prend implicitement en compte la durée typique d'une interférence, là où l'estimation par mesure cyclique ne dispose pas de telles informations et utilise une valeur fixe pour la durée d'une interférence. Deuxièmement, les mesures continues de bruit n'ont lieu qu'au début de l'expérience de mesure du PER vrai des liens, alors que l'histogramme basé sur les mesures cycliques regroupe une description du bruit perçu tout au long de l'expérience.

4.3.3.3 Évaluation de la précision de l'estimation du PER

La distribution de la précision de l'estimation du PER pour toutes les données enregistrées durant l'expérience est tracée en figure 4.14 en utilisant un graphe *letter-value* [112]. La valeur de la précision est calculée comme la différence entre le PER estimé et la borne la plus proche de l'intervalle de confiance à 95 % du PER mesuré sur chaque lien. Les valeurs sont groupées en plusieurs classes pour analyser l'effet des différents paramètres. Les 1re, 2e et 3e colonnes montrent les résultats sur plusieurs sites (Grenoble, Lille et Lyon). Les 3e, 4e et 5e colonnes montrent les résultats pour plusieurs longueurs de paquet (33, 84 et 126 octets). Les deux dernières montrent les résultats sur des canaux peu ou très chargés (en superposition ou non avec des canaux IEEE 801.11). Dans tous les cas, la valeur absolue de la précision est inférieure à 5 points de pourcentage pour la majeure partie des échantillons, ce qui montre que le modèle prend efficacement en compte ces paramètres.

4.3.3.4 L'effet du CCA

L'écoute du bruit radio lors des transmissions est une technique bien connue, dans le cadre des protocoles CSMA/CA (*Carrier Sense Multiple Access* / *Collision Avoidance*). L'écoute du bruit dans le modèle présenté dans cette section est différente, car elle est réalisée du côté du récepteur plutôt que du côté de l'émetteur. Réalisée côté émetteur, la mesure du bruit permet de détecter la présence d'interférence à cet instant, et éviter les collisions en retardant l'émission de la trame. L'intérêt est double : non seulement la trame sera émise plus tard, lorsque le niveau de bruit sera plus favorable, mais aussi l'interférence observée correspond fort probablement à une autre transmission en cours, qui ne sera du coup pas gênée par l'arrivée subite d'une autre trame — la collision a été évitée. Côté récepteur, l'instant d'arrivée du paquet est imprévisible et le récepteur ne peut de toute façon rien y changer. Le résultat de la mesure de bruit donne ainsi plutôt une indication statistique de la probabilité de réception — ce que le modèle décrit ici cherche à fournir. Les niveaux de bruit côté émetteur et côté récepteur sont bien sûr corrélés, mais pas nécessairement identiques, par exemple, à cause du problème de terminal caché. De plus, le CCA (*Clear Channel Assessment*) n'est effectué que juste avant le début de la transmission. Ainsi, il ne protège pas la trame contre une interférence apparaissant pendant la transmission, soit parce que l'émetteur ne réalise pas de CCA (c'est le cas des fours à micro-ondes p. ex.) ou qui ne détecte pas la transmission IEEE 802.15.4 lancée (parce que sa bande passante en fréquence est supérieure ou parce que son seuil de détection du bruit est trop élevé).

Durant les expériences, le CCA a été désactivé pour éviter les perturbations. Pour les couches liaison de données qui utilisent le CCA (ce qui n'est pas toujours le cas, comme par exemple pour certaines variantes de TSCH), l'estimation peut conduire vraisemblablement à surestimer les pertes car l'émetteur pourrait renoncer à transmettre du fait de la présence d'un signal sur le canal. Après l'avoir mesurée en activant le CCA dans d'autres expérimentations, cette surestima-

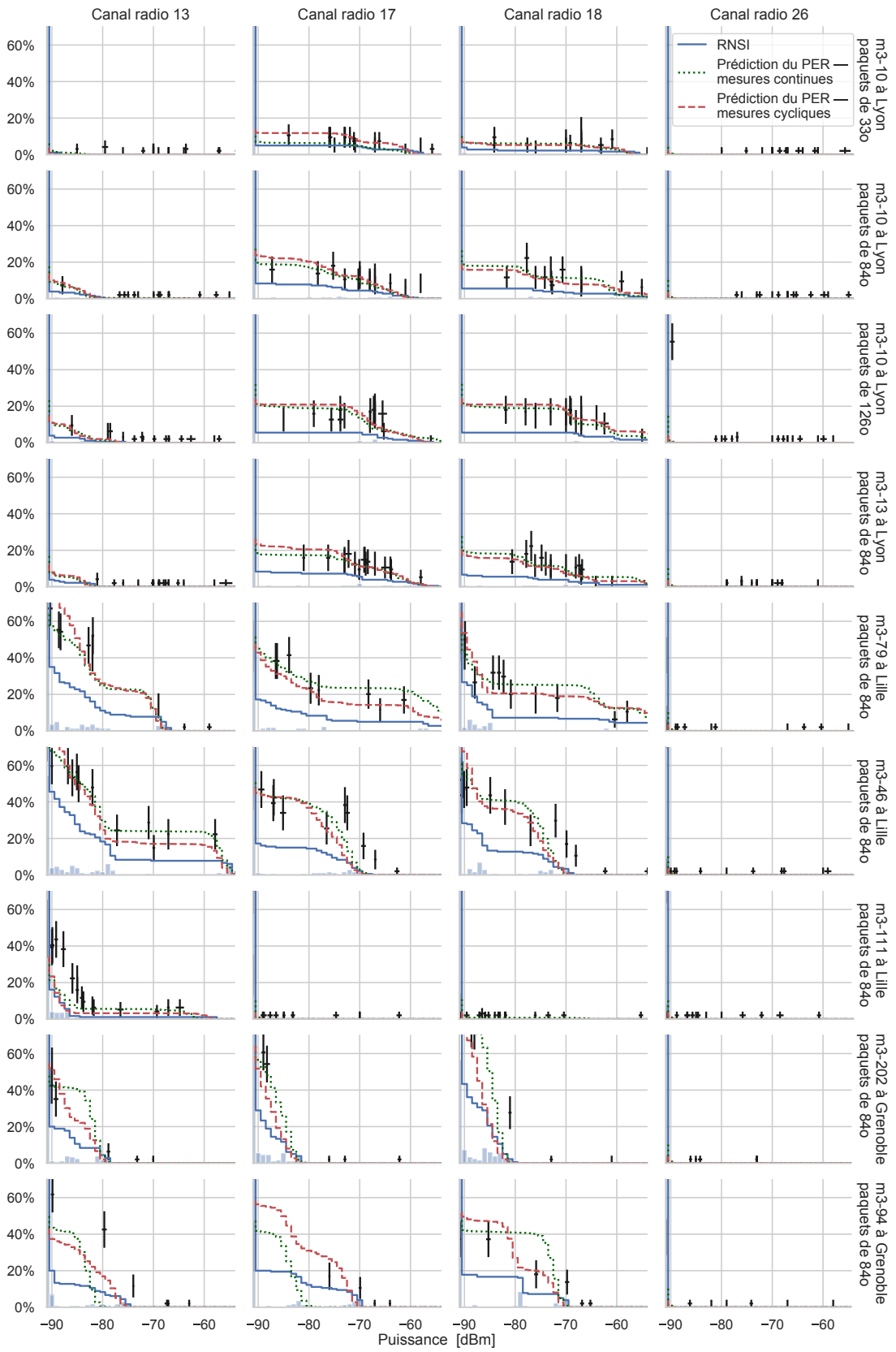


FIGURE 4.13 – Représentation du bruit du canal et de la qualité des liens (mesurée et prédite), par capteur récepteur (ligne) et par canal (colonne). L'histogramme et sa CCDF (ligne continue) représente la distribution du RNSI, mesurée par le capteur. Les croix représentent les liens depuis d'autres capteurs : la largeur représente l'écart-type du RSSI observé sur ce lien et la hauteur représente l'intervalle de confiance à 95 % du PER vrai. Les deux lignes en pointillés représentent l'estimation du PER avec les deux méthodes de mesure du bruit décrites ici.

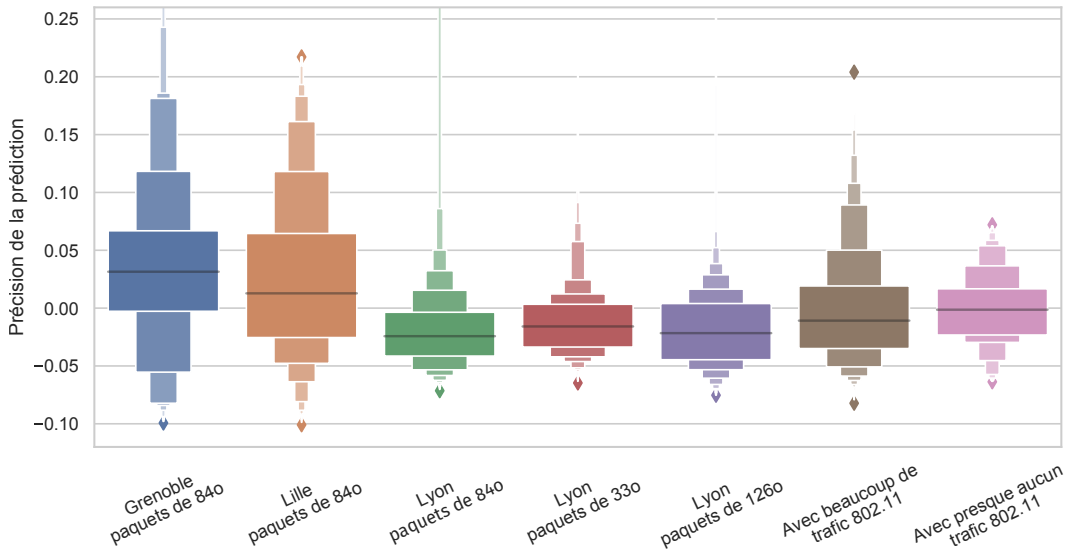


FIGURE 4.14 – Distribution (graphique *letter-value*) de la précision de l'estimation du PER c'est-à-dire la différence entre le PER estimé et la borne la plus proche de l'intervalle de confiance à 95 % du PER mesuré sur chaque lien.

tion s'est révélée diminuer le PER de moins de 10 % pour 97 % des liens, ce qui est comparable à la précision de l'estimation (cf. section 4.3.3.3). Ainsi, l'activation du CCA ne joue que légèrement sur l'estimation du PER.

Conclusion

Ce chapitre a donc présenté plusieurs techniques d'estimation rapide du PER. Ces techniques sont présentées accompagnées d'expérimentations afin de les valider. Elles sont basées sur diverses mesures physiques :

— **à partir de la puissance du signal sur le lien** (RSSI). Cette méthode offre des résultats fiables sur les liens pour lesquels la puissance du signal à la réception est élevée — nettement plus élevée que le bruit ambiant ; mais il est impossible de différencier les bons des mauvais liens lorsque la puissance approche du seuil de sensibilité radio, même de plusieurs ordres de grandeur.

— **à partir de l'indicateur de qualité des liens** calculé par la radio (LQI) — calculé ici à partir de corrélation de plusieurs symboles. Cet indicateur capture bien l'effet de l'interférence perçue par le paquet lors de sa transmission, et donne ainsi des résultats fiables pour les canaux où le niveau d'interférence est faible. Mais le LQI donne des résultats discutables sur des canaux plus chargés où le niveau d'interférence auquel le paquet devra faire face est variable. De plus cette méthode est sujette à des questions de portabilité, car le calcul du LQI n'est pas énoncé précisément dans la norme et l'implémentation n'est pas identique sur toutes les plateformes.

— **à partir d'une mesure double, du niveau de bruit et de la puissance du signal**. Cette méthode permet une estimation précise du PER. Elle nécessite l'échange de très peu de paquets (un seul dans des environnements relativement immobiles), qui peuvent aisément être des paquets choisis, utiles par exemple pour le routage, ce qui la rend apte à gérer des liens peu utilisés ou nouvellement découverts. Elle offre des résultats précis tant sur des petits que sur

des longs paquets, tant sur des canaux fréquentiels avec beaucoup de bruit que sur des canaux silencieux. La procédure de collecte des données est parfaitement adaptée aux réseaux de capteurs sans fil à faible énergie et petites capacités de calcul.

L'estimation du PER est faite du côté récepteur, où le RSSI est disponible ainsi que la distribution du bruit. Le protocole de routage doit encore mettre cette mesure à la disposition des voisins avant toute décision de routage, afin de gérer l'asymétrie des liaisons. Lorsque des acquittements sont attendus, le PER dans les deux directions peuvent être combinés pour calculer une valeur d'ETX sur ce lien.

Dans certains cas, une seule trame n'est pas suffisante pour évaluer correctement la puissance de réception des paquets d'un voisin. Dans le cas des canaux de Rayleigh à trajets multiples présentant une certaine mobilité (soit des capteurs eux-mêmes, soit d'objets dans leur environnement), le gain de canal global peut varier considérablement. Cette modification du gain du canal donne lieu à de plus grandes variations dans l'intensité du brouillage — bien que cette variation soit captée par la distribution du bruit — mais aussi et surtout dans le RSSI des voisins. De telles variations ont déjà été rapportées dans plusieurs études dans des situations réelles [85]. Dans ce cas, quelques paquets supplémentaires deviennent nécessaire pour déterminer la stabilité du RSSI d'un voisin, afin de calculer une gamme de valeurs de RSSI possibles et de les combiner avec la distribution du bruit.

Conclusion générale

L'objet d'étude de cette thèse est le routage pour les réseaux de capteurs sans fil. Le trafic attendu dans ces réseaux correspond principalement à des données générées par les capteurs et extraites hors du réseau. Le support de communication est un support sans fil de caractéristiques assez variables. Les capteurs eux-mêmes sont des plateformes légères, dont la puissance de calcul et les réserves énergétiques sont limitées, et où la radio est une ressource à économiser.

Les modes d'économie d'énergie proposés par la norme (et nécessaires pour prolonger la durée de vie des batteries) impliquent par effet de bord un coût plus élevé pour les messages multidiffusés, qui sont alors à éviter. Or c'est seulement l'inondation — et, à l'échelle d'un lien, la multidiffusion — qui permet de distribuer une information dans le réseau lorsque la topologie du réseau est inconnue. Pour organiser les nœuds entre eux et pouvoir ensuite offrir un jeu de routes pour acheminer les données, le protocole de routage est fatalement obligé d'inonder le réseau ; seulement, il s'agit de le faire de façon efficace, afin de réduire cette surcharge à son seul nécessaire, et construire efficacement une représentation de la topologie sur laquelle le protocole de routage doit travailler.

Comme nous venons de le voir dans le dernier chapitre, les efforts de routage seront sapés à la base si cette représentation ne décrit pas correctement les caractéristiques de transmission sur les liens disponibles. J'ai ainsi travaillé sur une mesure du taux de prévisionnel de paquets perdus sur un lien sans fil. Cette mesure est basée sur une mesure double, du niveau de bruit et de la puissance du signal, et des expérimentations pratiques l'ont montrée représentative de la réalité. Elle est tout à fait adaptée aux contraintes des réseaux de capteurs dont nous parlons, et rapide à effectuer — idéalement sur la réception d'un unique message — permettant ainsi d'être utilisée avec confiance sur des liens récemment découverts ou rarement utilisés. Sur une telle représentation, LRP peut alors construire un jeu de routes pour acheminer le trafic du réseau de capteurs, en respectant les exigences fixées pour un tel protocole : il organise les nœuds sous forme d'arbre de collecte qui permet d'extraire le trafic montant prédominant tout en fournissant aussi des routes d'hôtes pour permettre le contrôle des capteurs ; de plus, il réagit efficacement aux modifications

de topologie inévitables en réparant efficacement les routes installées, en n'affectant qu'un petit nombre de nœuds du réseau ; enfin, il offre une technique de validation du chemin emprunté par chaque paquet de données afin de garantir la détection des boucles avant leur utilisation.

La conception de ces mécanismes a systématiquement été pilotée autant que validée par des expérimentations pratiques. Ce sont ces expérimentations sur le protocole de routage qui nous ont fait remarquer la faiblesse de la métrique *hop count* que nous utilisions, et étudier les caractéristiques de transmission des liens pour proposer une mesure de qualité adaptée. Il serait intéressant, afin de boucler la boucle, de réaliser des expérimentations de routage en utilisant cette métrique.

Les normes, le matériel de plus en plus performant, sécurisé, et proche des besoins du marché, ainsi que les logiciels développés aujourd'hui permettent la mise en place d'applications viables industriellement, comme des réseaux de surveillance de structures industrielles (les compteurs Linky d'Enedis déployés dans la grille de distribution d'électricité, ou les SmartMesh de la société Analog Devices, anciennement Dust Networks, ou encore des réseaux de surveillance de plantations pour la protection contre le gel). Les réseaux de capteurs sans fil tels que présentés dans cette thèse ne sont certes pas les seuls à fournir des services de collecte d'information générée par des mesures physiques. Les réseaux LPWAN (*Low-Power Wide Area Networks*) en sont un autre exemple, avec des débits très faibles et des portées très étendues. Mais le modèle de trafic attendu est différent. Les LPWAN sont conçus pour collecter de petites quantités d'information dans des espaces très étendus, permettant ainsi des applications de type alerte ou de type positionnement ponctuel d'objets. Ça n'est pas la solution adaptée lorsque l'information à collecter est présente en trop grand volume, ou dans des environnements avec beaucoup d'obstacles pouvant impliquer le masquage des transmissions à une partie du réseau, où une solution multisaut est alors préférable. Dans certains scénarios précis, il serait alors intéressant de combiner ces deux approches, avec un réseau multisaut autonome à courte portée, qui conserverait la possibilité de remonter des données agrégées et simplifiées ou des alarmes à travers un réseau à longue portée.

Bibliographie

Références personnelles

[1] H.-J. AUDÉOUD, M. KRÓL, M. HEUSSE et A. DUDA. “Low overhead Loop-free Routing in Wireless Sensor Networks”. In : *11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE. Octobre 2015. DOI : 10.1109/wimob.2015.7347996. URL : http://hal.univ-grenoble-alpes.fr/hal-01286291/document.

[2] H.-J. AUDÉOUD et M. HEUSSE. “Vers un protocole de routage de surcharge minimale dans les réseaux de capteurs sans fils”. In : *18èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (ALGOTEL)*. Mai 2016. URL : https://hal.archives-ouvertes.fr/hal-01302991/document.

[3] H.-J. AUDÉOUD et M. HEUSSE. “Experimental Comparison of Routing Protocols for Wireless Sensors Networks : Routing Overhead and Asymmetric Links”. In : *29th International Teletraffic Congress (ITC)*. Septembre 2017. DOI : 10.23919/itc.2017.8064339. URL : http://hal.univ-grenoble-alpes.fr/hal-01609451/document.

[4] H.-J. AUDÉOUD et M. HEUSSE. “Quick and Efficient Link Quality Estimation in Wireless Sensors Networks”. In : *14th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*. IEEE. Février 2018. DOI : 10.23919/wons.2018.8311667. URL : https://hal.archives-ouvertes.fr/hal-01796508/document.

[5] H.-J. AUDÉOUD. *Implémentation du protocole de routage LRP sous Linux*. URL : https://github.com/audeoudh/pylrp.

[6] H.-J. AUDÉOUD, M. HEUSSE et C.-A. LA. *Implémentation du protocole de routage LRP sous Contiki*. URL : https://github.com/drakkar-lig/contiki/tree/lrp/core/net/lrp.

Documents & normes de l’IEEE

[7] *Coexistence of Wireless Personal Area Networks with Other Wireless Devices Operating in Unlicensed Frequency Bands. IEEE Std 802.15.2-2003*. Institute of Electrical et Electronics Engineers (IEEE), 2003.

[8] *IEEE Standard for Low-Rate Wireless Networks. IEEE Std 802.15.4-2006*. Institute of Electrical et Electronics Engineers (IEEE), 2006.

[9] *Coexistence analysis of IEEE Std 802.15.4 with other IEEE standards and proposed standards*. Institute of Electrical et Electronics Engineers (IEEE), septembre 2010.

[10] *IEEE Standard for Low-Rate Wireless Networks. IEEE Std 802.15.4-2015.* Institute of Electrical et Electronics Engineers (IEEE), 2015.

[11] *IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Std 802.11-2016.* Institute of Electrical et Electronics Engineers (IEEE), 2016.

Documents & normes de l'IETF

[12] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *Routing Information Protocol.* RFC 1058. Internet Engineering Task Force, juin 1988. DOI : 10.17487/RFC1058.

[13] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *OSI IS-IS Intra-domain Routing Protocol.* RFC 1142. Internet Engineering Task Force, février 1990. DOI : 10.17487/RFC1142.

[14] Phillip G. GROSS. *Choosing a Common IGP for the IP Internet.* RFC 1371. Internet Engineering Task Force, octobre 1992. DOI : 10.17487/RFC1371.

[15] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *A Border Gateway Protocol 4 (BGP-4).* RFC 1771. Internet Engineering Task Force, mars 1995. DOI : 10.17487/RFC1771.

[16] Randy BUSH et Robert ELZ. *Serial Number Arithmetic.* RFC 1982. Internet Engineering Task Force, août 1996. DOI : 10.17487/RFC1982.

[17] Bob HINDEN et Dr. Steve E. DEERING. *Internet Protocol, Version 6 (IPv6) Specification.* RFC 2460. Internet Engineering Task Force, décembre 1998. DOI : 10.17487/RFC2460.

[18] Gary S. MALKIN. *RIP Version 2.* RFC 2453. Internet Engineering Task Force, novembre 1998. DOI : 10.17487/RFC2453.

[19] John MOY. *OSPF Version 2.* RFC 2328. Internet Engineering Task Force, avril 1998. DOI : 10.17487/RFC2328.

[20] Rob COLTUN, Dennis FERGUSON et John MOY. *OSPF for IPv6.* RFC 2740. Internet Engineering Task Force, décembre 1999. DOI : 10.17487/RFC2740.

[21] Thomas H. CLAUSEN et Philippe JACQUET. *Optimized Link State Routing Protocol (OLSR).* RFC 3626. Internet Engineering Task Force, octobre 2003. DOI : 10.17487/RFC3626.

[22] Samir R. DAS, Charles E. PERKINS et Elizabeth M. BELDING-ROYER. *Ad hoc On-Demand Distance Vector (AODV) Routing.* RFC 3561. Internet Engineering Task Force, juillet 2003. DOI : 10.17487/RFC3561.

[23] Yakov REKHTER, Susan HARES et Tony LI. *A Border Gateway Protocol 4 (BGP-4).* RFC 4271. Internet Engineering Task Force, janvier 2006. DOI : 10.17487/RFC4271.

[24] Yih-Chun HU, Dave A. MALTZ et David B. JOHNSON. *The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4.* RFC 4728. Internet Engineering Task Force, février 2007. DOI : 10.17487/RFC4728.

[25] Gabriel MONTENEGRO, Jonathan HUI, David CULLER et Nandakishore KUSHALNAGAR. *Transmission of IPv6 Packets over IEEE 802.15.4 Networks.* RFC 4944. Internet Engineering Task Force, septembre 2007. DOI : 10.17487/RFC4944.

[26] William A. SIMPSON, Dr. Thomas NARTEN, Erik NORDMARK et Hesham SOLIMAN. *Neighbor Discovery for IP version 6 (IPv6).* RFC 4861. Internet Engineering Task Force, septembre 2007. DOI : 10.17487/RFC4861.

[27] Dave KATZ et David WARD. *Bidirectional Forwarding Detection (BFD)*. RFC 5880. Internet Engineering Task Force, juin 2010. DOI : 10.17487/RFC5880.

[28] Juliusz CHROBOCZEK. *The Babel Routing Protocol*. RFC 6126. Internet Engineering Task Force, avril 2011. DOI : 10.17487/RFC6126.

[29] Thomas H. CLAUSEN, Christopher DEARLOVE et Justin DEAN. *Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP)*. RFC 6130. Internet Engineering Task Force, avril 2011. DOI : 10.17487/RFC6130.

[30] P LEVIS, Thomas H. CLAUSEN, Omprakash GNAWALI, Jonathan HUI et JeongGil KO. *The Trickle Algorithm*. RFC 6206. Internet Engineering Task Force, mars 2011. DOI : 10.17487/RFC6206.

[31] Pascal THUBERT et Jonathan HUI. *Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks*. RFC 6282. Internet Engineering Task Force, septembre 2011. DOI : 10.17487/RFC6282.

[32] Roger ALEXANDER et al. *RPL : IPv6 Routing Protocol for Low-Power and Lossy Networks*. RFC 6550. Internet Engineering Task Force, mars 2012. DOI : 10.17487/RFC6550.

[33] Stuart CHESHIRE et Marc KROCHMAL. *Multicast DNS*. RFC 6762. Internet Engineering Task Force, février 2013. DOI : 10.17487/RFC6762.

[34] Thomas H. CLAUSEN, Christopher DEARLOVE, Philippe JACQUET et Ulrich HERBERG. *The Optimized Link State Routing Protocol Version 2*. RFC 7181. Internet Engineering Task Force, avril 2014. DOI : 10.17487/RFC7181.

[35] Zach SHELBY, Klaus HARTKE et Carsten BORMANN. *The Constrained Application Protocol (CoAP)*. RFC 7252. Internet Engineering Task Force, juin 2014. DOI : 10.17487/RFC7252.

[36] Donnie SAVAGE, James NG, Steven MOORE, Donald SLICE, Peter PALUCH et Russ WHITE. *Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP)*. RFC 7868. Internet Engineering Task Force, mai 2016. DOI : 10.17487/RFC7868.

[37] Xavier VILAJOSANA, Kris PISTER et Thomas WATTEYNE. *Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration*. RFC 8180. Internet Engineering Task Force, mai 2017. DOI : 10.17487/RFC8180.

[38] Qin WANG, Xavier VILAJOSANA et Thomas WATTEYNE. *6TiSCH Operation Sublayer (6top) Protocol (6P)*. RFC 8480. Internet Engineering Task Force, novembre 2018. DOI : 10.17487/RFC8480.

[39] Vincent D. PARK et Dr. Scott M. CORSON. *Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification*. Internet-Draft draft-ietf-manet-tora-spec-04. Work in Progress. Internet Engineering Task Force, juillet 2001.

[40] Zygmunt HAAS, Marc R. PEARLMAN et Prince SAMAR. *The Zone Routing Protocol (ZRP) for Ad Hoc Networks*. Internet-Draft draft-ietf-manet-zone-zrp-04. Work in Progress. Internet Engineering Task Force, août 2002.

[41] Abhay ROY. *Adjacency Reduction in OSPF using SPT Reachability*. Internet-Draft draft-roy-ospf-smart-peering-01. Work in Progress. Internet Engineering Task Force, novembre 2005.

[42] Richard OGIER et Phil SPAGNOLO. *MANET Extension of OSPF using CDS Flooding*. Internet-Draft draft-ogier-manet-ospf-extension-10. Work in Progress. Internet Engineering Task Force, novembre 2007.

[43] Abhay ROY et M CHANDRA. *Extensions to OSPF to Support Mobile Ad Hoc Networking*. Internet-Draft draft-chandra-ospf-manet-ext-05. Work in Progress. Internet Engineering Task Force, août 2007.

[44] Axel NEUMANN, Corinna AICHELE, Marek LINDNER et Simon WUNDERLICH. *Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.)* Internet-Draft draft-wunderlich-openmesh-manet-routing-00. Work in Progress. Internet Engineering Task Force, avril 2008.

[45] Ahmed AYADI, David ROS et Laurent TOUTAIN. *TCP header compression for 6LoWPAN*. Internet-Draft draft-aayadi-6lowpan-tcphc-01. Work in Progress. Internet Engineering Task Force, octobre 2010.

[46] Colin O'FLYNN. *ICMPv6/ND Compression for 6LoWPAN Networks*. Internet-Draft draft-oflynn-6lowpan-icmphc-00. Work in Progress. Internet Engineering Task Force, juillet 2010.

[47] Thomas H. CLAUSEN et al. *The LLN On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng)*. Internet-Draft draft-clausen-lln-loadng-04. Work in Progress. Internet Engineering Task Force, avril 2012.

[48] Mukul GOYAL, Dominique BARTHEL et Emmanuel BACCELLI. *DIS Modifications*. Internet-Draft draft-goyal-roll-dis-modifications-01. Work in Progress. Internet Engineering Task Force, octobre 2012.

[49] Thomas H. CLAUSEN et al. *The Lightweight On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng)*. Internet-Draft draft-clausen-lln-loadng-15. Work in Progress. Internet Engineering Task Force, juillet 2016.

Articles scientifiques

[50] E. B. WILSON. "Probable Inference, the Law of Succession, and Statistical Inference". In : *Journal of the American Statistical Association* 22 (1927).

[51] R. V. L. HARTLEY. "Transmission of Information". In : *Bell System technical journal* 7 (juin 1928). DOI : 10.1002/j.1538-7305.1928.tb01236.x.

[52] L. R. Jr FORD. *Network flow theory*. Rapp. tech. Rand Corp Santa Monica CA, août 1956.

[53] R. BELLMAN. "On a Routing Problem". In : *Quarterly of applied mathematics* (1958). DOI : 10.1090/qam/102435.

[54] E. W. DIJKSTRA et C. S. SCHOLTEN. "Termination Detection for Diffusing Computations". In : *Information Processing Letters* (1980).

[55] R. PERLMAN. "Fault-Tolerant Broadcast of Routing Information". In : *Computer Networks* 7 (1983). DOI : 10.1016/0376-5075(83)90034-X.

[56] J. J. GARCIA-LUNES-ACEVES. "Loop-Free Routing Using Diffusing Computations". In : *Transactions on Networking* (février 1993). DOI : 10.1109/90.222913.

[57] D. B. JOHNSON. "Routing in Ad Hoc Networks of Mobile Hosts". In : *First Workshop on Mobile Computing Systems and Applications*. Décembre 1994. DOI : 10.1109/WMCSA.1994.33.

[58] C. E. PERKINS et P. BHAGWAT. "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers". In : *SIGCOMM Comput. Commun. Rev.* 24 (octobre 1994). DOI : 10.1145/190809.190336.

[59] D. B. JOHNSON et D. A. MALTZ. “Dynamic Source Routing in Ad hoc Wireless Networks”. In : *Mobile Computing*. Boston, MA : Springer US, 1996. DOI : 10.1007/978-0-585-29603-6_5.

[60] Z. J. HAAS. “A new routing protocol for the reconfigurable wireless networks”. In : *6th International Conference on Universal Personal Communications (ICUPC)*. IEEE. 1997. DOI : 10.1109/ICUPC.1997.627227.

[61] V. D. PARK et M. S. CORSON. “A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks”. In : *INFOCOM*. IEEE. 1997. DOI : 10.1109/INFCOM.1997.631180.

[62] C. E. PERKINS et E. M. ROYER. “Ad-hoc On-demand Distance Vector Routing”. In : *Second Workshop on Mobile Computing Systems and Applications (WMCSA)*. Février 1999. DOI : 10.1109/MCSA.1999.749281.

[63] A. EL-HOIYDI. “Aloha with preamble sampling for sporadic traffic in ad hoc wireless sensor networks”. In : *International Conference on Communications (ICC)*. T. 5. IEEE. 2002. DOI : 10.1109/ICC.2002.997465.

[64] A. WOO et D. CULLER. *Evaluation of Efficient Link Reliability Estimators for Low-Power Wireless Networks*. Rapp. tech. EECS Department, University of California, Berkeley, 2003.

[65] A. WOO, T. TONG et D. CULLER. “Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks”. In : *In SenSys*. 2003.

[66] A. DUNKELS, B. GRONVALL et T. VOIGT. “Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors”. In : *29st Conference on Local Computer Networks*. IEEE. Novembre 2004.

[67] J. HASSAN et S. JHA. “Optimising Expanding Ring Search for Multi-Hop Wireless Networks”. In : *Global Telecommunications Conference (GLOBECOM)*. T. 2. IEEE. 2004. DOI : 10.1109/vtcfall.2012.6399334.

[68] M. ZUNIGA et B. KRISHNAMACHARI. “Analyzing the Transitional Region in Low Power Wireless Links”. In : *SECON Conference*. 2004.

[69] Z. CHENG et W. B. HEINZELMAN. “Flooding Strategy for Target Discovery in Wireless Networks”. In : *Wireless Networks* 11 (2005). DOI : 10.1007/s11276-005-3516-7.

[70] D. S. J. DE COUTO, D. AGUAYO, J. BICKET et R. MORRIS. “A High-throughput Path Metric for Multi-hop Wireless Routing”. In : *Wireless Networks* 11 (juillet 2005).

[71] J. DENG. “Locating Randomly Selected Destinations in Large Multi-hop Wireless Networks”. In : *19th International Teletraffic Congress (ITC-19)*. 2005. URL : https://www.uncg.edu/cmp/faculty/j_deng/papers/ers_itc19.pdf.

[72] S. HARA et al. “Propagation characteristics of IEEE 802.15.4 radio signal and their application for location estimation”. In : *61st Vehicular Technology Conference*. IEEE. 2005.

[73] F. OSTERLIND, A. DUNKELS, J. ERIKSSON, N. FINNE et T. VOIGT. “Cross-Level Sensor Network Simulation with COOJA”. In : *31st Conference on Local Computer Networks*. IEEE. Novembre 2006.

[74] P. A. SPAGNOLO et T. R. HENDERSON. “Comparison of Proposed OSPF MANET Extensions”. In : *Military Communications conference (MILCOM)*. IEEE. Octobre 2006. DOI : 10.1109/MILCOM.2006.302376.

[75] K. SRINIVASAN, P. DUTTA, A. TAVAKOLI et P. LEVIS. “Understanding the Causes of Packet Delivery Success and Failure in Dense Wireless Sensor Networks”. In : *4th International Conference on Embedded Networked Sensor Systems (SenSys)*. 2006. DOI : 10.1145/1182807.1182885.

[76] P. MARINA, W. LILI, M. PETRI et R. JANNE. “Interference Measurements on Performance Degradation between Colocated IEEE 802.11g/n and IEEE 802.15.4 Networks”. In : *Sixth International Conference on Networking (ICN)*. 2007.

[77] S. Y. SHIN, H. S. PARK et W. H. KWON. “Mutual interference analysis of IEEE 802.15.4 and IEEE 802.11b”. In : *Computer networks* 51 (2007). DOI : 10.1016/j.comnet.2007.01.034.

[78] M. Z. ZAMALLOA et B. KRISHNAMACHARI. “An Analysis of Unreliability and Asymmetry in Low-Power Wireless Links”. In : *ACM Transactions on Sensor Networks* (juin 2007).

[79] L. ANGRISANI, M. BERTOCCO, D. FORTIN et Sona A. “Experimental Study of Coexistence Issues Between IEEE 802.11b and IEEE 802.15.4 Wireless Networks”. In : *IEEE Transactions on Instrumentation and Measurement* 57 (août 2008).

[80] M. ZENNARO, H. NTAREME et A. BAGULA. “Experimental Evaluation of Temporal and Energy Characteristics of an Outdoor Sensor Network”. In : *Mobility Conference*. ACM Press, 2008.

[81] O. GNAWALI, R. FONSECA, K. JAMIESON, D. MOSS et P. LEVIS. “Collection Tree Protocol”. In : *7th ACM Conference on Embedded Networked Sensor Systems*. 2009. DOI : 10.1145/1644038.1644040.

[82] N. TSIFTES, A. DUNKELS, Z. HE et T VOIGT. “Enabling Large-Scale Storage in Sensor Networks with the Coffee File System”. In : *International Conference on Information Processing in Sensor Networks*. IEEE. Août 2009. ISBN : 978-1-60558-371-6.

[83] T. CLAUSEN et U. HERBERG. “Comparative Study of RPL-Enabled Optimized Broadcast in Wireless Sensor Networks”. In : *6th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*. Décembre 2010. DOI : 10.1109/ISSNIP.2010.5706795.

[84] L. SHU, Y. ZHANG, L. T. YANG, Y. WANG, M. HAUSWIRTH et N. XIONG. “TPGF : geographic routing in wireless multimedia sensor networks”. In : *Telecommunication Systems* 44 (2010). DOI : 10.1007/s11235-009-9227-0.

[85] K. SRINIVASAN, P. DUTTA, A. TAVAKOLI et P. LEVIS. “An Empirical Study of Low-power Wireless”. In : *ACM Transactions on Sensor Networks* 6 (2010). DOI : 10.1145/1689239.1689246.

[86] N. TSIFTES, J. ERIKSSON et A. DUNKELS. “Low-power wireless IPv6 routing with ContikiRPL”. In : *9th International Conference on Information Processing in Sensor Networks*. ACM/IEEE. 2010.

[87] N. BACCOUR, A. KOUBÂA, D. JAMÂA M. B. do Rosário, H. YOUSSEF, M. ALVES et L. B. BECKER. “RadiaLE : A Framework for Designing and Assessing Link Quality Estimators in Wireless Sensor Networks”. In : *Ad Hoc Networks* 9 (septembre 2011).

[88] T. H. CLAUSEN et U. HERBERG. *Some Considerations on Routing In Particular and Lossy Environments*. Rapp. tech. INRIA, 2011. URL : https://hal.inria.fr/inria-00565830/.

[89] T. CLAUSEN, U. HERBERG et M. PHILIPP. “A Critical Evaluation of the “IPv6 Routing Protocol for Low Power and Lossy Networks” (RPL)”. In : *7th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE. Octobre 2011. DOI : 10.1109/wimob.2011.6085374.

[90] A. DUNKELS. *The ContikiMAC Radio Duty Cycling Protocol*. Rapp. tech. Swedish Institute of Computer Science, 2011.

[91] N. BACCOUR et al. "Radio Link Quality Estimation in Wireless Sensor Networks : A Survey". In : *ACM Transactions on Sensor Networks* 8 (2012).

[92] T. CLAUSEN, J. YI et A. COLIN DE VERDIERE. "LOADng : Towards AODV Version 2". In : *Vehicular Technology Conference (VTC Fall)*. IEEE. 2012. DOI : 10.1016/j.comnet.2017.06.025.

[93] J. YI, T. CLAUSEN et A. BAS. "Smart Route Request for On-demand Route Discovery in Constrained Environments". In : *International Conference on Wireless Information Technology and Systems (ICWITS)*. IEEE. 2012. DOI : 10.1109/icwits.2012.6417755.

[94] A. BILDEA. "Link Quality in Wireless Sensor Networks". Thèse de doct. Université de Grenoble, novembre 2013.

[95] K. HEURTEFEUX, H. MENOUAR et N. ABUALI. "Experimental evaluation of a Routing Protocol for WSNs : RPL robustness under study". In : *9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE. Octobre 2013. DOI : 10.1109/WiMOB.2013.6673404.

[96] O. IOVA, F. THEOLEYRE et T. NOEL. "Stability and efficiency of RPL under realistic conditions in Wireless Sensor Networks". In : *24th International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*. IEEE. 2013. DOI : 10.1109/PIMRC.2013.6666490.

[97] C.-A. LA, M. HEUSSE et A. DUDA. "Link Reversal and Reactive Routing in Low Power and Lossy Networks". In : *24th International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*. IEEE. Juin 2013.

[98] M. VUČINIĆ, B. TOURANCHEAU et A. DUDA. "Performance comparison of the RPL and LOADng routing protocols in a home automation scenario". In : *Wireless Communications and Networking Conference (WCNC)*. IEEE. 2013. DOI : 10.1109/WCNC.2013.6554867.

[99] J. YI, T. CLAUSEN et Y. IGARASHI. "Evaluation of routing protocol for low power and Lossy Networks : LOADng and RPL". In : *Conference on Wireless Sensor (ICWISE)*. IEEE. 2013. DOI : 10.1109/ICWISE.2013.6728773.

[100] J. BROWN, U. ROEDIG, C. A. BOANO et K. RÖMER. "Estimating Packet Reception Rate in Noisy Environments". In : *39th Conference on Local Computer Networks Workshops*. IEEE. 2014. DOI : 10.1109/LCNW.2014.6927706.

[101] J. TRIPATHI, J. C. DE OLIVEIRA et J.-P. VASSEUR. "Proactive versus reactive routing in low power and lossy networks : Performance analysis and scalability improvements". In : *Ad Hoc Networks* 23 (2014).

[102] J. YI et T. CLAUSEN. "Collection Tree Extension of Reactive Routing Protocol for Low-Power and Lossy Networks". In : *International Journal of Distributed Sensor Networks* (2014). DOI : 10.1155/2014/352421.

[103] A. YUSHEV, P. LEHMANN et A. SIKORA. "6LoWPAN with RPL performance measurements in an Automated Physical Testbed". In : *2nd International Sympsium on Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems : Technology and Applications (IDAACS-SWS)*. Septembre 2014. DOI : 10.1109/IDAACS-SWS.2014.6954618.

[104] Julien COUTINHO. *Réparation locale du routage dans les réseaux de capteurs*. Rapport de recherche. Laboratoire d'Informatique de Grenoble, mai 2015. URL : https://ensiwiki.ensimag.fr/index.php?title=Julien_Coutinho_(avec_Martin_Heusse)_:_R%C3%A9paration_locale_du_routage_dans_les_r%C3%A9seaux_de_capteurs (visité le 09/10/2019).

[105] S. ELYENGUI, R. BOUHOUCHI et T. EZZEDINE. "A Comparative Performance Study of the Routing Protocols RPL, LOADng and LOADng-CTP with Bidirectional Traffic for AMI Scenario". In : *International Conference on Intelligent Computer Communication and Processing (ICCP)*. IEEE. 2015. DOI : 10.1109/ICCP.2015.7312719.

[106] E. FLEURY et al. "FIT IoT-LAB : The Largest IoT Open Experimental Testbed". In : *ERCIM News* (2015).

[107] F. VAN TRIMPONT, G. B. KATUMBA, V. MOEYAERT et S. BETTE. "Impact of the weak link count mechanism on G3-PLC LOADng routing protocol". In : *International Symposium on Power Line Communications and Its Applications (ISPLC)*. IEEE. Mars 2015. DOI : 10.1109/ISPLC.2015.7147598.

[108] L.-O. VARGA et al. "GreenNet : an Energy Harvesting IP-enabled Wireless Sensor Network". In : *IEEE Internet of Things Journal* (2015).

[109] K. BRUN-LAGUNA, A. L. DIEDRICHS, D. DUJOVNE, R. LÉONE, X. VILAJOSANA et T. WATTEYNE. "(Not so) Intuitive Results from a Smart Agriculture Low-Power Wireless Mesh Deployment". In : *11th Workshop on Challenged Networks*. ACM. 2016. DOI : 10.1145/2979683.2979696.

[110] M. MAMDOUH, K. ELSAYED et A. KHATTAB. "RPL Load Balancing via Minimum Degree Spanning Tree". In : *12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE. Décembre 2016. DOI : 10.1109/WiMOB.2016.7763185.

[111] T. CLAUSEN, J. YI et U. HERBERG. "Lightweight On-demand Ad hoc Distance-vector Routing - Next Generation (LOADng) : Protocol, extension, and applicability". In : *Computer Networks* 126 (2017). DOI : 10.1016/j.comnet.2017.06.025.

[112] H. HOFMANN, H. WICKHAM et K. KAFADAR. "Value Plots : Boxplots for Large Data". In : *Journal of Computational and Graphical Statistics* 26 (2017). DOI : 10.1080/10618600.2017.1305277.

[113] E. MORIN, M. MAMAN, R. GUIZZETTI et A. DUDA. "Comparison of the Device Lifetime in Wireless Networks for the Internet of Things". In : *IEEE Access* 5 (novembre 2017). DOI : 10.1109/ACCESS.2017.2688279.

[114] G. Z. PAPADOPOULOS, A. GALLAIS, G. SCHREINER, E. JOU et T. NOEL. "Thorough IoT testbed Characterization : from Proof-of-concept to Repeatable Experimentations". In : *Elsevier Computer Networks* 119 (2017).

[115] S. FU, Y. ZHANG, M. CERIOTTI, Y. JIANG, M. PACKEISER et P. J. MARRÓN. "Modeling Packet Loss Rate of IEEE 802.15.4 Links in Diverse Environmental Conditions". In : *Wireless Communications and Networking Conference (WCNC)*. IEEE. Avril 2018. DOI : 10.1109/WCNC.2018.8377111.

[116] Y. ZHANG, S. FU, Y. JEANG, M. CERIOTTI, M. PACKEISER et P. J. MARRÓN. "An LQI-Based Packet Loss Rate Model for IEEE 802.15.4 Links". In : *International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*. Septembre 2018.

Documentations techniques

[117] *CC2520 datasheet : 2.4 GHz IEEE 802.15.4 / Zigbee RF Transceiver*. Texas Instruments. Décembre 2007.

[118] *AT86RF231/ZU/ZF datasheet*. Atmel. Octobre 2009.

[119] *CC2538 System-on-Chip Solution for 2.4-GHz IEEE 802.15.4 and Zigbee/Zigbee IP Applications*. Version C. Texas Instruments. Avril 2015.

[120] *CC2420 datasheet : 2.4 GHz IEEE 802.15.4 / Zigbee RF Transceiver*. Rev. C. Texas Instruments. Avril 2017.

En ligne

[121] *Routing Over Low power and Lossy networks. Groupe de travail de l'IETF*. 2008. URL : https://datatracker.ietf.org/wg/roll/charter/ (visité le 09/10/2019).

[122] *IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH). Groupe de travail de l'IETF*. 2013. URL : https://datatracker.ietf.org/wg/6tisch/charter/ (visité le 09/10/2019).

[123] FIT/IoT-LAB. *M3 Open Node motes*. 27 avril 2018. URL : https://github.com/iot-lab/iot-lab/wiki/Hardware-M3-node (visité le 09/10/2019).

[124] Berkeley, université de Californie. *OpenWSN project*. URL : https://openwsn.atlassian.net/wiki/spaces/OW/overview (visité le 09/10/2019).

[125] *Contiki-NG : The OS for Next Generation IoT Devices*. URL : https://www.contiki-ng.org/ (visité le 09/10/2019).

[126] Docker, Inc. *Docker : Entreprise Container Platform*. URL : https://www.docker.com (visité le 09/10/2019).

[127] FIT/IoT-LAB. *IoT-LAB : a very large scale open testbed*. URL : www.iot-lab.info (visité le 09/10/2019).

[128] Freifunk. *B.A.T.M.A.N. (better approach to mobile ad-hoc networking)*. URL : https://www.open-mesh.org/projects/open-mesh/wiki (visité le 09/10/2019).

[129] G3-PLC Alliance. *Enabling the smartest grid... Together*. URL : http://www.g3-plc.com/home/ (visité le 09/10/2019).

[130] Netfilter Core Team. *The netfilter.org project*. URL : https://netfilter.org/ (visité le 09/10/2019).

[131] OpenConnectivity Fundation. *IoTvity*. URL : https://iotivity.org/about (visité le 09/10/2019).

[132] Thread Group. *OpenThread, an Open-Source Implementation of the Thread Networking Protocol*. URL : https://github.com/openthread/openthread (visité le 09/10/2019).

[133] Thread Group. *Thread : A New Wireless Networking Protocol for the Home*. URL : http://threadgroup.org/ (visité le 09/10/2019).

[134] Zigbee Alliance. *Dotdot*. URL : https://zigbee.org/zigbee-for-developers/dotdot/ (visité le 09/10/2019).

[135] Zigbee Alliance. *Zigbee 3.0*. URL : https://zigbee.org/zigbee-for-developers/zigbee-3-0/ (visité le 09/10/2019).

Autres références

[136] EURO-COST231, éd. *Urban transmission loss models for mobile radio in the 900 and 1 800 MHz bands*. Version 2. Septembre 1991.

[137] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *Intermediate System to Intermediate System intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service (ISO 8473)*. Version 2. ISO/IEC 10589:2002. Novembre 2002.

[121] *Routing Over Low power and Lossy networks. Groupe de travail de l'IETF*. 2008. URL : https://datatracker.ietf.org/wg/roll/charter/ (visité le 09/10/2019).

[122] *IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH). Groupe de travail de l'IETF*. 2013. URL : https://datatracker.ietf.org/wg/6tisch/charter/ (visité le 09/10/2019).

[138] PARLEMENT EUROPÉEN et CONSEIL EUROPÉEN. *Directive 2014/53/UE*. Avril 2014.

[139] ITU-T. *G.9903 : Narrowband orthogonal frequency division multiplexing power line communication transceivers for G3-PLC networks*. Août 2017.

[123] FIT/IOT-LAB. *M3 Open Node motes*. 27 avril 2018. URL : https://github.com/iot-lab/iot-lab/wiki/Hardware-M3-node (visité le 09/10/2019).

[124] BERKELEY, UNIVERSITÉ DE CALIFORNIE. *OpenWSN project*. URL : https://openwsn.atlassian.net/wiki/spaces/OW/overview (visité le 09/10/2019).

[125] *Contiki-NG : The OS for Next Generation IoT Devices*. URL : https://www.contiki-ng.org/ (visité le 09/10/2019).

[126] DOCKER, INC. *Docker : Entreprise Container Platform*. URL : https://www.docker.com (visité le 09/10/2019).

[127] FIT/IOT-LAB. *IoT-LAB : a very large scale open testbed*. URL : www.iot-lab.info (visité le 09/10/2019).

[128] FREIFUNK. *B.A.T.M.A.N. (better approach to mobile ad-hoc networking)*. URL : https://www.open-mesh.org/projects/open-mesh/wiki (visité le 09/10/2019).

[129] G3-PLC ALLIANCE. *Enabling the smartest grid... Together*. URL : http://www.g3-plc.com/home/ (visité le 09/10/2019).

[130] NETFILTER CORE TEAM. *The netfilter.org project*. URL : https://netfilter.org/ (visité le 09/10/2019).

[131] OPENCONNECTIVITY FUNDATION. *IoTvity*. URL : https://iotivity.org/about (visité le 09/10/2019).

[132] THREAD GROUP. *OpenThread, an Open-Source Implementation of the Thread Networking Protocol*. URL : https://github.com/openthread/openthread (visité le 09/10/2019).

[133] THREAD GROUP. *Thread : A New Wireless Networking Protocol for the Home*. URL : http://threadgroup.org/ (visité le 09/10/2019).

[134] ZIGBEE ALLIANCE. *Dotdot*. URL : https://zigbee.org/zigbee-for-developers/dotdot/ (visité le 09/10/2019).

[135] ZIGBEE ALLIANCE. *Zigbee 3.0*. URL : https://zigbee.org/zigbee-for-developers/zigbee-3-0/ (visité le 09/10/2019).