N° d'ordre : 04/2009–M/IN

République Algérienne Démocratique et Populaire
Ministère de L'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumdien
Faculté d'Électronique et d'Informatique

MÉMOIRE
Présenté pour l'obtention du diplôme de MAGISTER
En Informatique
Spécialité : Informatique Mobile

Par : OUADJAOUT Abdelraouf

Sujet

Fiabilité de Dissémination dans les Réseaux de Capteurs Sans Fils

Soutenu le 04/01/2009, devant le jury composé de :

Mr- N. BADACHE,	Professeur,	USTHB,	Président
Mr- Y. CHALLAL,	Docteur,	U.T. COMPIEGNE,	Directeur de Thèse
Mr- A. AISSANI,	Professeur,	USTHB,	Examinateur
Mr- D. TANDJAOUI,	Chargé de Recherche,	CERIST,	Invité

Remerciements

Louanges à Allâh Seigneur des mondes, et que la paix et le salut soient sur notre Prophète Mohamed, l'Envoyé en clémence pour l'univers, ainsi que sur sa famille, et sur tous ses compagnons.

Je remercie énormément Dr. Yacine Challal, mon directeur de thèse, pour m'avoir donné la possibilité de travailler sur ce sujet de recherche extrêmement intéressant. Je le remercie pour son aide précieuse et ses conseils, qui m'ont permis de bien comprendre les différentes facettes de ce sujet et de bien analyser les problèmes posés.

Je tiens aussi à remercier vivement les membres du Jury : Prof. Nadjib Badache, Prof. Amar Aissani et Dr. Djamel Tandjaoui, pour le temps et l'effort investis pour juger mon travail. Je remercie spécialement le Prof. Badache pour m'avoir donné l'occasion d'effectuer un stage scientifique à l'EPFL, qui m'a beaucoup aidé pour accomplir ce travail et bien comprendre le domaine des capteurs sans fil.

Je remercie énormément mes très chers parents et mes frères, qui m'ont soutenu vivement durant toutes mes études. Leur amour et leurs conseils m'ont été extrêmement précieux durant toutes ces années.

Je voudrais aussi remercier mes amis de post-graduation avec qui j'ai partagé d'excellents moments. Toute ma gratitude à Noureddine Lasla et Miloud Bagaa, qui m'ont beaucoup aidé avec leurs analyses pertinentes et leur précieux soutien.

Enfin, je voudrais remercier vivement les membres du laboratoire LCA1 à l'EPFL pour leur accueil chaleureux et leur aide durant mon stage à Lausanne. Je remercie spécialement Prof. Jean-Pierre Hubaux qui m'a chaleureusement accueilli au sein de son groupe et qui m'a permis de travailler sur des problèmes très intéressants. Je voudrais aussi exprimer mes sincères remerciements à Jacques Panchard et Reza Shokri, qui m'ont permis d'apprendre énormément sur ce domaine et qui m'ont sans cesse soutenu durant ce stage.

Résumé

Les réseaux de capteurs sans fil représentent un domaine très prometteur. Cette technologie permet d'offrir une grande variété d'applications dans divers domaines, avec une grande facilité de déploiement et un faible coût de production. Cependant, ce type de systèmes est sujet à diverses formes de défaillances qui affectent négativement la fiabilité du réseau. Ces défaillances regroupent les pannes matérielles *ordinaires*, mais aussi les attaques informatiques qui représentent de nos jours une menace réelle sans cesse croissante. À cause de la sensibilité de nombreuses applications des réseaux de capteurs, plusieurs travaux de recherche ont été menés afin de trouver des solutions permettant de faire fonctionner ce genre de réseaux en présence de pannes et d'intrus.

Dans ce travail, on présente une nouvelle solution de dissémination, appelée SEIF (*Secure and Efficient Intrusion-Fault tolerant routing protocol for wireless sensor networks*), tolérante aux intrus et aux pannes, et offrant un grand niveau de fiabilité grâce à une topologie multi-chemins sécurisée. Contrairement aux solutions existantes, notre protocole repose sur un mécanisme distribué de construction et de détection, qui ne requiert aucune communication avec la station de base pour la vérification des chemins découverts. En plus, SEIF introduit une nouvelle variante de sélection des chemins alternatifs visant à améliorer la tolérance de la topologie, tout en conservant l'énergie des nœuds. Ce travail présente aussi une nouvelle taxonomie des solutions existantes, ainsi qu'une évaluation de performances avec simulations. Pour cela, on a utilisé l'environnement TinyOS, le système *de facto* des réseaux de capteurs. Les résultats obtenus confirment l'efficacité de notre approche par rapport aux protocoles les plus représentatifs de la littérature.

Durant ce Magistère, nous avons publié les travaux suivants :

[1] A. Ouadjaout, Y. Challal, A. Bachir, N. Lasla, M. Bagaa, and L. Khelladi. "Information Security in Wireless Sensor Networks". In *Encyclopedia of Ad Hoc and Ubiquitous Computing*. Editor D. Agrawal & B. Xie, World Scientific Publishers, 2009. To appear.

[2] A. Ouadjaout, Y. Challal, N. Lasla, and M. Bagaa. "SEIF : Secure and Efficient Intrusion-Fault tolerant routing protocol for wireless sensor networks". In *the Third International Conference on Availability, Reliability and Security (IEEE ARES '08)*, pages 503–508, Spain, March 2008.

[3] J. Panchard, E. Costanza, J. Freudiger, A. Ouadjaout, B. Bostanipour, and J.-P. Hubaux. "Making the Invisible Audible : Acoustic Interfaces for the Management of Wireless Sensor Networks". Technical Report, Ecole Polytechnique Fédérale de Lausanne, Switzerland, June 2008.

[4] M. Bagaa, N. Lasla, A. Ouadjaout, and Y. Challal. "SEDAN : Secure and Efficient Data Aggregation protocol for wireless sensor Networks". In *the 32nd IEEE Conference on Local Computer Networks (LCN '07)*, pages 1053–1060, Ireland, October 2007.

Table des matières

Introduction

Depuis leur création, les réseaux de communication sans fil ont connu un succès sans cesse croissant au sein des communautés scientifiques et industrielles. Grâce à ses divers avantages, cette technologie a pu s'instaurer comme étant un acteur incontournable dans les architectures réseaux actuelles. Ce média hertzien offre en effet des propriétés *uniques*, qui peuvent être résumées en trois points : *la facilité du déploiement*, *l'ubiquité de l'information* et *le coût réduit d'installation*.

Au cours de son évolution, le paradigme sans fil a vu naître diverses architectures dérivées, telles que : les réseaux cellulaires, les réseaux locaux sans fils et autres. Durant cette dernière décennie, une architecture nouvelle a vu le jour : *les réseaux de capteurs sans fil*. Ce type de réseaux résulte d'une fusion de deux pôles de l'informatique moderne : les systèmes embarqués et les communications sans fil. Un réseau de capteurs sans fil, ou WSN (*Wireless Sensor Network*), est composé d'un ensemble d'unités de traitements embarquées, appelées *motes*, communiquant via des liens sans fil [1]. Le but général d'un WSN est la collecte d'un ensemble de paramètres de l'environnement entourant les motes, telles que la température ou la pression de l'atmosphère, afin de les acheminer vers des points de collecte, appelés *stations de base* (ou SB).

Les WSN sont souvent considérés comme étant les successeurs des réseaux *ad hoc*. En effet, les WSN partagent avec les MANET (*Mobile Ad hoc NETworks*) plusieurs propriétés en commun, telles que l'absence d'infrastructure et les communications sans fil. Mais l'une des différences clés entre les deux architectures est *le domaine d'application*. Contrairement aux réseaux MANET, qui n'ont pas pu connaître un vrai succès, les WSN ont pu attirer un nombre croissant d'industriels, vu leur *réalisme* et leur apport *concret*. En effet, le besoin d'un suivi continu d'un environnement donné est assez courant dans multiples activités de la société. Les processus industriels, les applications militaires de tracking, le monitoring d'habitat, ainsi que l'agriculture de précision ne sont que quelques exemples d'une panoplie vaste et variée d'applications possibles du suivi continu offert par les WSN. Grâce à ce potentiel riche en applications, les WSN on su se démarquer de leur origine MANET et attirer de grandes firmes à travers le monde, telles que IBM, Sun, Intel et Philips.

Malheureusement, *les WSN ne sont pas parfaits* ! À cause de leur faible coût et leur

déploiement dans des zones parfois hostiles, les motes sont assez *fragiles* et vulnérables à diverses formes de *défaillances* : cassure, épuisement de la batterie, ... *etc.* Ces problèmes rendent les WSN des systèmes à *fragilité innée*, ce qui doit être considéré comme une propriété *normale* du réseau.

Néanmoins, les pannes physiques « *ordinaires* » ne sont pas la seule cause des défaillances dans ce genre de système. La courte histoire des systèmes de communication a démontré que ce genre d'environnements (*i.e.* les système de communication) a toujours été une proie privilégiée d'attaques informatiques qui ne cessent de s'amplifier. La dernière étude annuelle du CSI (*Computer Security Institute*), menée sur 194 organisations, a montré que les pertes financières causées par les attaques informatiques ont atteint le total de 66 930 950$, avec une moyenne de 345 005$ par organisation [3]. Il est important de noter que la moyenne de l'année passée était de 167 713$, ce qui montre une croissance étonnante. Les WSN étant des systèmes de communications assez fragiles, ils représentent donc des proies faciles pour les attaquants, motivés par diverses raisons : politiques, militaires, économiques ou bien par simple curiosité.

L'un des facteurs importants pouvant aggraver l'ampleur d'une défaillance d'un WSN est *la valeur de la donnée véhiculée*. En effet, les WSN peuvent être utilisés dans des applications *critiques* où la donnée captée possède une valeur d'une grande importance. Dans ce cas, cette donnée doit impérativement parvenir aux utilisateurs finaux sans *altération* ni *manipulation* par des entités externes. Par conséquent, une attention particulière a été focalisée sur la notion de *fiabilité* dans les WSN par plusieurs groupes de recherche à travers le monde.

Les différentes études menées sur la fiabilité dans les WSN couvrent toutes les couches de communication : de la couche physique jusqu'aux applications. Toutefois, la couche de routage a bénéficié d'une attention particulière à cause de son importance vitale au sein des systèmes de communication saut par saut. A travers ces travaux, les chercheurs ont essayé de trouver des solutions permettant de faire fonctionner un WSN en présence d'éventuelles défaillances matérielles ou attaques d'intrus. Cependant, ils ont été confronté à un challenge intéressent, qui est de trouver *le bon compromis* entre le niveau de fiabilité offert et le coût (principalement en énergie) nécessaire pour y parvenir. Parmi les solutions existantes, aucun protocole ne permet d'obtenir un compromis *intéressant* entre ces deux objectifs.

Dans ce travail, nous avons proposé un nouveau protocole, appelé SEIF (*Secure and Efficient Intrusion-Fault tolerant routing protocol for wireless sensor networks*) [27], permettant d'offrir un tel compromis. SEIF permet de construire une topologie de communication sécurisée très tolérante aux pannes et aux intrus, ne nécessitant qu'un échange très réduit de messages. Grâce à sa nature totalement distribuée, les attaques d'intrus sont instantanément détectées et stoppées par les motes. En plus, SEIF propose une nouvelle variante de topologie de communication permettant d'augmenter considérablement la durée de vie du réseau même en présence de pannes.

Ce travail sera organisée comme suit. Le chapitre 1 décrira l'état de l'art des architectures matérielles et logicielles des WSN actuels. Cet état de l'art présentera aussi quelques applications réelles profitant des avantages des WSN afin de résoudre des problèmes concrets

de la société. Le chapitre 2 présentera une taxonomie nouvelle des solutions de fiabilité de dissémination dans les WSN. Cette taxonomie décrit les différentes vulnérabilités au niveau de la couche de routage, leurs causes ainsi que les objectifs à réaliser afin de les empêcher. Dans le chapitre 3, on décrira en détail les différentes solutions existantes citées dans notre taxonomie. Après description des inconvénients des approches étudiées, le chapitre 4 présente notre protocole SEIF, ainsi que les résultats de simulations effectués avec TinyOS.

Chapitre 1

Les Réseaux de Capteurs Sans Fil : Un État de l'Art

1.1 Introduction

La technologie des capteurs sans fil connaît actuellement une évolution remarquable sur les plans logiciels et matériels. Grâce aux divers débouchés possibles, une vive concurrence s'est instaurée pour promouvoir ce domaine, que ce soit au sein des communautés industrielles ou scientifiques. Certains analystes ont même désigné cette technologie comme étant l'une des dix technologies qui vont changer le monde [40] !

L'une des clés du succès de cette technologie est la simplicité de son architecture matérielle. Actuellement, on peut dire que le marché matériel des WSN est assez mûr, et peut répondre aux exigences d'une grande partie des applications actuelles. En contre partie, l'aspect logiciel quant à lui souffre d'une certaine instabilité. En effet, la communauté scientifique s'est beaucoup focalisée sur cette thématique, produisant un nombre considérable de prototypes et de protocoles. Cette richesse est certes essentielle pour le développement continu de la technologie, mais elle entraîne une absence d'interopérabilité entre les différents composants du système. Or, cette interopréabilité est vitale pour le succès commercial de la technologie.

La suite de ce chapitre présentera les principales architectures logicielles et matérielles disponibles dans le marché des WSN actuels. On enchaînera ensuite par la présentation des différents composants logiciels qu'on peut retrouver dans des réseaux de capteurs actuels ou futurs. Enfin, on terminera par la présentation de quelques exemples d'utilisation réelles de réseaux de capteurs déployés par des industriels.

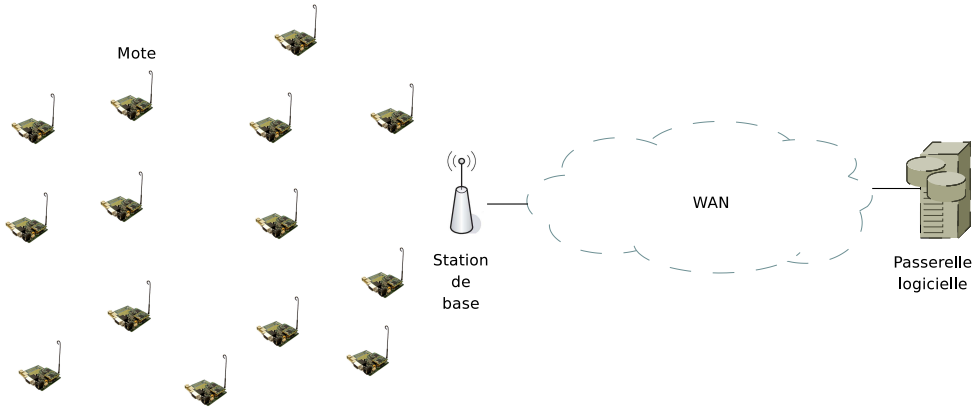


Fig. 1.1: Un réseau de capteurs sans fil

1.2 Architecture matérielle

La figure 1.1 schématise un réseau de capteurs typique et ses différents composants. Dans cette architecture, on distingue trois entités : *les motes*, *la station de base* et *la passerelle logicielle*.

1.2.1 Les motes

Un mote est une entité de traitement miniaturisée dotée d'une interface de communication sans fil et fonctionnant avec une source énergétique limitée.

L'unité de traitement est constituée d'un micro-contrôleur à faible consommation d'énergie. Les motes les plus répandus actuellement se basent principalement sur deux marques :

- La série ATMega128L d'Atmel est présente dans les motes Mica2 et MicaZ de Crossbow. Ce micro-contrôleur 8-bits fonctionne avec une vitesse de 8 MHz. Il est doté de 128Ko de Flash et 4Ko de RAM.
- La famille MSP430 de Texas Instrument est une autre alternative qu'on retrouve dans les TinyNode de Shockfish, les Tmote Sky et les BTNodes. Le MSP430 se base sur une architecture 16 bits, tourne sous 8 MHz, et possède 48Ko de Flash et 10Ko de RAM.

Récemment, une nouvelle génération de motes a été introduite, caractérisée par une plus grande puissance. Les SPOT de Sun sont un exemple de cette nouvelle génération. Ils sont dotés d'un processeur ARM920T 32 bits, fonctionnant avec 180 MHz, 512Ko de RAM et 4Mo de Flash. Mais vu leur prix actuel, on préfère encore utiliser des architectures plus simples et donc moins coûteuses.

Pour créer le réseau de communication, le micro-contrôleur est lié à une interface réseau sans fil à l'aide d'un bus SPI. Vu la spécificité des WSN, ces interfaces réseau sans fil sont

caractérisées par une faible consommation d'énergie et un faible débit. Dans un but de normaliser ce type de communications, l'IEEE a développé le standard 802.15.4 [41]. Cette norme définit les détails de la couche physique, ainsi que la méthode d'accès au média. Actuellement, il existe plusieurs firmes produisant des circuits conformes à cette norme. La radio CC2420 de Texas Instruments[1] est le composant le plus utilisé actuellement, qu'on retrouve dans les motes MicaZ, SunSPOT, Tmote Sky et autres. Toutefois, l'utilisation de cette norme n'est pas encore totalement généralisée, car il existe certains industriels fournissant des motes qui opèrent sur un modèle radio non standard. Par exemple, les motes Mica2 utilisent la radio CC1100, fonctionnant entre 300 MHz et 1 GHz, ce qui est totalement incompatible avec le standard 802.15.4.

Le troisième composant d'un mote est sa source d'énergie. Cette dernière représente la ressource la plus importante dans un WSN. Cette importance est due au fait que ces réseaux doivent fonctionner d'une manière autonome dans des zones distantes, telles que des forêts, des champs agricoles, ... *etc.* Par conséquent, l'approvisionnement des nœuds en énergie devient un point crucial afin de faire fonctionner le réseau le plus longtemps possible.

Le moyen le plus commun d'approvisionnement est d'utiliser des piles traditionnelles. Cette solution souffrent d'inconvénients majeurs, particulièrement lorsque le réseau est déployé dans des zones à accès difficile, où le changement des piles n'est pas évident. Une autre solution consiste à extraire l'énergie à partir de l'environnement de déploiement afin d'assurer une alimentation continue. Cette technique, appelée *Energy Harvesting*, vise à transformer la lumière, la chaleur ou l'énergie cinétique disponibles dans l'environnement en énergie électrique. Comme ces sources sont *auto-renouvelables*, la durée de vie du réseau est considérablement améliorée. Entre ces différentes approches, l'énergie solaire reste l'approche la plus connue. L'énergie cinétique, quant à elle, n'a été introduite que récemment via des prototypes transformant les vibrations subites par le mote en énergie électrique [2]. Cela est extrêmement avantageux dans des applications où les motes sont en perpétuel mouvement, comme c'est le cas pour les applications véhiculaires.

Le micro-contrôleur, la radio et l'énergie sont les trois composants communs entre les différents types de motes. En rajoutant certains équipements, le mote revêtira un rôle bien précis au sein du réseau. On peut distinguer entre trois principaux rôles : les nœuds *capteurs*, les nœuds *relais* et les nœuds *actionneurs*.

1.2.1.1 Les nœuds capteurs

Un nœud capteur est un mote doté d'une (ou plusieurs) sonde(s) permettant de mesurer un phénomène caractérisant l'environnement de déploiement. Les fabricants de motes proposent généralement des cartes de capture permettant de récupérer des paramètres standards, tels que la température, la position GPS, la pression atmosphérique, ... *etc.* Cependant, pour certaines applications spécifiques, il est nécessaire de brancher des sondes externes produites

[1] CC2420 a été développé initialement par ChipCon, qui a été rachetée par la suite par TI.

par d'autres firmes, et de développer par la suite des pilotes afin de faire communiquer le micro-contrôleur et la sonde en question.

1.2.1.2 Les nœuds relais

Un nœud relais est en réalité un simple mote déployé pour assurer la connexité du réseau. Par conséquent, il peut être épargné de toute tâche de capture. Ainsi, chaque type de mote ayant une interface radio peut être considéré comme étant un nœud relais. Mais dans certaines situations de trafic important, la tâche de relais peut être assez lourde à supporter par un simple mote, vu que les communications radios sont l'opération la plus coûteuse en énergie. Donc certains nœuds relais peuvent être des « super motes » dotés de plus d'énergie et de puissance de traitement afin d'assurer correctement leur rôle.

1.2.1.3 Les nœuds actionneurs

Après la phase de relais, la donnée parvient aux utilisateurs finaux pour traitement via la passerelle logicielle. Suivant la nature de l'application, on peut avoir plusieurs scénarios. Dans un scénario basique, les données captées sont traitées puis archivées dans une base de données. Lors de la détection d'un événement particulier, l'action à entreprendre est effectuée *manuellement* sur terrain. Un exemple de ce type d'applications est la protection contre les feux de forêts, dans laquelle les pompiers doivent se déplacer au lieu de l'incendie. Un deuxième scénario plus avancé consiste à doter les motes par des circuits permettant de contrôler des objets extérieurs. Ces *nœuds actionneurs* peuvent donc assurer une réaction face à un changement de l'environnement. Par exemple, dans une application d'irrigation automatique, certains nœuds sont connectés à des vannes permettant de contrôler la quantité d'eau à libérer suivant l'humidité du sol captée.

1.2.2 La station de base

La station de base représente le puits vers lequel toutes les données seront dirigées. Avec ce rôle central, la station de base peut donc fonctionner comme passerelle réseau afin d'acheminer les données captées vers des réseaux distants accessibles par les utilisateurs finaux. Pour ce faire, plusieurs alternatives sont envisageables : GPRS, Wi-Fi, Wi-Max, ou Ethernet (si disponible). Dans le cas d'absence de toute infrastructure réseau (filaire ou pas), la station de base doit archiver localement les données reçues, ce qui permettra à l'administrateur réseau de les récupérer ultérieurement. Il faut noter que dans la plus part des cas, la station de base est un simple mote doté d'interfaces réseaux supplémentaires (le plus souvent GPRS ou Ethernet), et qui peut être alimentée d'une manière continue par prise.

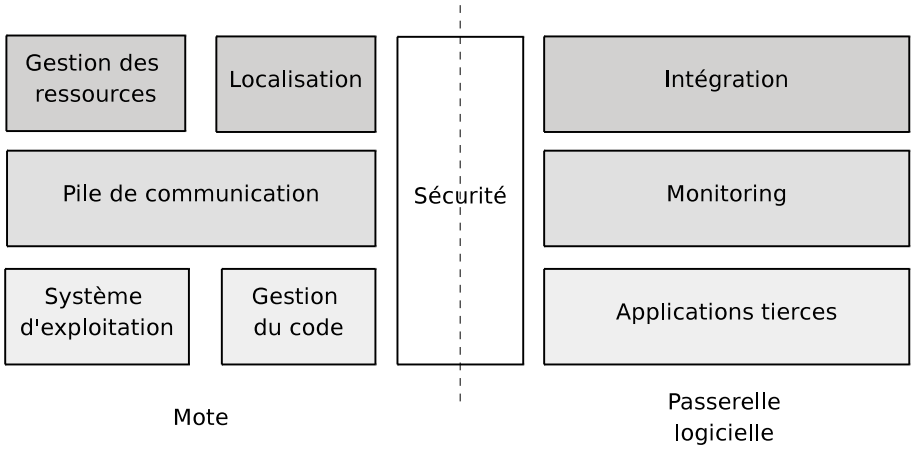


Fig. 1.2: Les principaux composants logiciels dans un WSN

1.2.3 La passerelle logicielle

Afin de faciliter l'exploitation d'un réseau de capteurs, plusieurs équipes de recherches ont essayé de développer des middlewares fournissant des abstractions d'accès au réseau physique et ses données. Ces middlewares offrent un ensemble de services afin d'interroger, actionner et gérer un réseau de capteurs distant. Plus de détails sur cet aspect seront donnés lors de la description de l'architecture logicielle.

1.3 Architecture logicielle

Vu le nombre considérable de travaux effectués autour des WSN, il est difficile de donner une architecture logicielle de référence couvrant tous les besoins des applications. Ce qui complique encore plus cette tâche est la diversité des disciplines informatiques qu'on peut retrouver dans un même WSN : les systèmes d'exploitation, les bases de données distribuées, les protocoles réseau, les algorithmes distribués, et tant d'autres. Toutes ces différentes technologies doivent être adaptées aux exigences et contraintes des WSN, ce qui représente en soi un défi de poids.

La figure 1.2 montre les composants logiciels les plus importants dans un WSN. Dans le reste de cette section, on présentera brièvement chaque composant et les principaux travaux développés.

1.3.1 Le système d'exploitation

Le système d'exploitation est la brique logicielle de base dans les WSN. Il permet de fournir un ensemble d'abstractions autour de la couche matérielle, tout en considérant les contraintes

et limitations des motes. Ces abstractions permettent d'une part, une plus grande portabilité du code pour des architectures matérielles hétérogènes, et d'autre part, la décomposition d'une application en modules pouvant être développés par des équipes distinctes.

Le composant principal d'un système d'exploitation est son scheduler. Ce dernier définit le mode d'exécution des processus et leur organisation au sein de la mémoire. Dans la littérature actuelle, il existe trois approches d'organisation de processus, qu'on présentera à travers un exemple illustratif de chaque famille.

La première approche est le *multithreading préemptif*, dans lequel le code exécutable est composé de threads qui peuvent s'exécuter séparément [33]. Puisque ces threads sont indépendants, chacun doit posséder sa propre pile d'exécution. Comme ce modèle est préemptif, permettant à un thread d'interrompre l'exécution d'un autre, les mécanismes de protections de variables partagées doivent être employés. Cette approche a été adoptée par le système LiteOS [33]. Autre que son scheduler avancé, LiteOS permet aussi de présenter le réseau de capteur d'une façon très originale. En effet, LiteOS modélise le réseau comme étant un système de fichier Unix. Ainsi, chaque nœud est vu comme un fichier virtuel, ce qui permet d'accéder aux différentes ressources du nœud (programmes, données, ... *etc*) à l'aide d'un shell similaire à ceux qu'on retrouve dans Unix, par des commandes telles que `cp`, `rm`, `cat`, ... *etc*.

L'inconvénient de l'approche multithread est qu'elle nécessite une pile pour chaque thread. Comme la taille de cette pile n'est pas connue a priori, elle doit être allouée au préalable avec une taille *surestimée*. Pour réduire cette consommation mémoire, TinyOS [13] a introduit le modèle événementiel. Dans ce modèle, le code est constitué d'un ensemble d'évènements nommés *tâches*. Une tâche est une fonction appelée pour être exécutée *ultérieurement*, et lorsqu'elle est choisie par le scheduler FIFO, aucune autre tâche ne peut l'interrompre jusqu'à sa terminaison. Par conséquent, le système ne contient qu'une seule tâche en cours d'exécution, ce qui enlève le besoin de gérer plusieurs pile en RAM. En plus du gain mémoire, la cohérence d'accès aux variables partagées est assurées entre les tâches elles-mêmes. Pour ajouter une concurrence d'exécution, ces tâches peuvent être interrompues par des évènements, généralement matériels, comme la réception d'un paquet radio. Pour éviter toute incohérence d'exécution, ces évènements doivent effectuer le moins de traitement possible, en mettant simplement à jour des variables d'états puis déposer des tâches ultérieures afin d'effectuer le traitement adéquat.

En analysant le modèle événementiel, on peut remarquer deux inconvénients :

- Lorsqu'une tâche est en cours d'exécution, elle ne peut être interrompue par une autre jusqu'à sa fin. Cela est acceptable lorsque l'application ne contient que des tâches « *légères* ». En présence de tâches lourdes nécessitant un grand temps de traitement (comme des opérations cryptographiques), la réactivité du système diminue grandement, ce qui inflige une limitation significative pour les systèmes où le temps de réponse est important.
- Le modèle événementiel pur ne supporte pas un mécanisme de blocage-attente suivant une certaine condition (par exemple, l'attente d'un acquittement). Ainsi, le traitement

avant et après la condition doivent être séparés dans deux portions de code exécutées asynchroniquement, ce qui nécessite l'utilisation de variables d'état globales afin de contrôler le flux d'exécution entre ces deux contextes d'exécution différents. L'ensemble de ces variables d'état se traduit donc à l'utilisation d'une machine à état. L'expérience a démontré que l'emploi d'une machine à état afin de gérer un flux d'exécution d'un programme d'évènements peut s'avérer difficile [23, 43].

Afin de résoudre ces deux problèmes, Adams et al. ont développé le système Contiki [7]. Contiki est un système événementiel, comme TinyOS, mais présentant deux caractéristiques importantes :

- Afin d'éviter le blocage du scheduler en cas de calculs intenses, Conitki permet l'utilisation optionnelle du modèle multithread. Cela est possible grâce une librairie optionnelle qui ne sera utilisée que pour les programmes qui en font appel explicitement. Ainsi, Conitki adopte un modèle hybride, qui est événementiel par défaut, mais qui peut être étendu en multithread à la demande.
- Pour faciliter le développement de code événementiel, tout en évitant l'abstraction par machine à état, Contiki introduit un nouveau paradigme de programmation : les *protothreads* [9]. Ce paradigme introduit un ensemble de primitives d'attente sur condition, ce qui enlève le besoin d'une programmation à travers une machine à état. Ainsi, on obtient un code séquentiel utilisant le même contexte d'exécution, et rendant le code plus lisible et plus court.

1.3.2 La pile réseau

L'essentiel de la littérature des WSN se focalise sur l'élaboration d'une pile protocolaire afin d'assurer une connexité correcte du réseau tout en préservant les ressources des motes. Ce composant, regroupant les primitives de communication de base, est une brique essentielle dans les réseau de capteurs. Cependant, en analysant le profil énergétique typique d'un nœud, on remarque que l'interface réseau est la source principale de consommation d'énergie. Par conséquent, l'optimisation du profil énergétique global passe forcément par l'optimisation de la pile réseau en l'adaptant aux propriétés des WSN.

1.3.2.1 La couche MAC

Le premier composant important de cette suite protocolaire est la couche MAC. Comme cette couche définit la politique d'accès au média sans fil – *i.e.* quand et comment le mote doit-il utiliser son interface sans fil – l'optimisation de la couche MAC permet de réduire énormément l'overhead de l'énergie. Pour cela, plusieurs politiques MAC ont été développées, dont le mécanisme général peut être classé en deux familles : les protocoles *synchrones* et les protocoles *asynchrones* [19].

Les protocoles MAC synchrones se basent sur une coordination des nœuds afin d'établir un schedule commun régissant l'accès au média [32, 10]. En utilisant des mécanismes dérivés de TDMA, ces protocoles peuvent éviter (ou éliminer) les collisions, mais aussi éviter les écoutes radio inutiles. Ce dernier point est essentiel, car le schedule commun permet d'*orchestrer* une communication totalement organisée et ordonnée dans le temps. Cela permet aux nœuds d'éteindre leurs interfaces sans fil périodiquement, ce qui économise efficacement l'énergie restante.

L'inconvénient des approches asynchrones est la nécessité d'échanger des messages de contrôle afin d'établir et maintenir le schedule commun. Pour y remédier, les protocoles asynchrones autorisent les nœuds à fonctionner d'une manière indépendante sans trop de complexité de coordination. Généralement, ce genre de protocoles se basent sur des méthodes probabilistes telles que CSMA/CA. Ainsi, les nœuds n'ont pas besoin de maintenir un état commun, ce qui réduit le nombre de messages de contrôle et facilite l'introduction de nouveaux nœuds au réseau.

1.3.2.2 Le routage

La fonction de routage est un bloc essentiel dans toute architecture de communication multi-sauts. Elle permet d'assurer la connexité logique entre des nœuds distants physiquement, même dans un environnement dynamique. Ce maintient se fait grâce à un relais saut par saut effectué via des chemins, formant ainsi une topologie de routage. Dans le domaine des WSN, on peut distinguer entre deux types de topologies :

- *Les topologies plusieurs-à-un*, ou de collection, sont les plus communes dans les WSN. Dans ce genre de paradigme, les motes collectent les données captées et les transmettent vers la SB. Comme tous les nœuds partagent la SB comme unique destination, la construction des routes est initiée par la SB. Cela s'effectue en utilisant une diffusion dans tout le réseau d'un message RREQ (*Route REQuest*). Ce message est relayé saut par saut afin de couvrir chaque nœud du réseau. Par conséquent, chaque mote va découvrir *proactivement* une route vers la SB à travers l'un de ses voisins, appelé *parent*. Cela résultera donc en une topologie en arbre, ayant la SB comme racine.

- *Les topologies mesh* se basent sur une communication point-à-point entre les nœuds du réseaux, ce qui est le paradigme le plus commun dans les réseaux ad hoc traditionnels. La construction des routes dans ce genre de topologies est initiée *réactivement* à la demande des capteurs. Quand un mote veut communiquer et ne possède pas de route fraîche, il inonde le réseau avec un message RREQ. Lorsque ce message atteint la destination (comme la SB par exemple), cette dernière répond avec une message RREP (*Route REPly*), qui va empreinter le chemin inverse du message RREQ jusqu'à atteindre la source. Il faut noter que la nature point-à-point des topologies mesh n'est pas adéquate au paradigme des WSN, thème de discussion de la fin du chapitre 4.

Sur ces différentes topologies, plusieurs mécanismes de relais de données peuvent être considérés. Une grande partie des travaux sur les WSN proposent des schémas propriétaires non-compatibles avec les infrastructures existantes. Chaque proposition adopte son propre format de paquets et ses propres messages de signalisation. Par conséquent, pour pouvoir connecter ce genre de réseaux au réseaux traditionnels, tels que Internet, des proxy doivent être déployés servant de *traducteurs* entre les deux mécanismes de relais (qui sont IP et celui du WSN).

Afin de faciliter encore plus cette intégration, plusieurs groupes de recherches et industriels travaillent sur l'accommodation d'une version de TCP/IP pour les WSN. En faisant fonctionner les nœuds eux mêmes sur IP, le réseau de capteur se voit donc automatiquement atteignable via Internet, et cela d'une manière tout à fait transparente. En plus, la pile TCP/IP est considérée comme une architecture *mature*, qui a fait ses preuves depuis plusieurs décennies. Cette maturité se traduit par une stabilité prouvée, ainsi que l'existence d'une pléthore de protocoles et d'outils d'administration découlant de besoins réels.

Dans un effort de standardisation, l'IETF a formé deux groupes de travail : 6LoWPAN et RoLL. Le groupe de travail 6LoWPAN (*IPv6 over Low power Wireless Personal Area Networks*) vise à adapter les paquets IPv6 afin de répondre aux exigences et limites des réseaux de capteurs. Le but du groupe RoLL (*Routing over Low power and Lossy networks*) est de fournir un framework commun de routage pour l'établissement des chemins et le choix des métriques adéquates à ce genre d'environnement.

1.3.3 La gestion du code

Comme tout système informatique, les erreurs de programmation et les éventuelles améliorations et optimisations sont souvent incontournables. Pour cela, un réseau de capteurs opérationnel doit prendre en considération le mécanisme de mise à jour du code des nœuds déployés. Sans gestion intelligente, l'administrateur devra recourir à l'approche traditionnelle manuelle, ce qui peut être extrêmement coûteux.

En effet, la méthode traditionnelle consiste à utiliser un accès physique au mote en injectant le nouveau code via le port de programmation du micro-contrôleur. Or, le réseau peut contenir un nombre considérable de nœuds, et dont le lieu de déploiement peut être difficilement accessible. Pour cela, des méthodes de déploiement distant de code ont été développées [14, 38, 44, 15]. L'administrateur peut injecter des mises à jour du code déployé en diffusant le nouveau code vers tout le réseau via un relais saut par saut. Chaque nœud va s'auto-reprogrammer avec la nouvelle version, épargnant à l'administrateur d'utiliser la méthode manuelle pour chaque mote du réseau.

Il faut noter que la méthode de reprogrammation dépend étroitement du système d'exploitation sous-jacent. La conception de TinyOS fait que cette reprogrammation ne peut être que totale, i.e. l'administrateur doit envoyer l'image complète du système, ainsi que toutes les applications, même si une application seulement a été modifiée. En contre partie, Conitki

permet une reprogrammation partielle du mote grâce à son mécanisme de chargement sélectif d'application en mémoire, ce qui économisera la bande passante nécessaire pour le transfert du nouveau code.

1.3.4 Gestion des ressources et monitoring

Le suivi de l'état d'un réseau de capteurs après son déploiement est une tâche essentielle pour la garantie d'un bon contrôle de la durée de vie du réseau. La principale ressource à monitorer est l'énergie des nœuds. En maintenant une vue précise sur cette métrique et en l'utilisant ingénieusement, la durée de vie du réseau peut être améliorée de 52% [29].

Malheureusement, l'obtention de cette information dans le contexte des WSN n'est pas triviale [16]. D'une part, les motes actuels n'offrent qu'une évaluation du voltage de la batterie, ce qui ne représente pas une estimation fiable de l'énergie restante. D'autre part, le profil énergétique des motes est souvent caractérisé par des fluctuations dans le temps, ce qui est dû à deux causes principales :

- Les composants matériels présentent des profils énergétiques extrêmement contrastés. La figure 1.3 montre le suivi de la consommation d'énergie d'un mote lors de l'exécution d'une application WSN typique, dont le fonctionnement est comme suit. L'application commence à activer son capteur de température et lit sa valeur (1.5s – 2.5s). Ensuite, l'interface radio est activée pour une durée d'une seconde afin de relayer les paquets des voisins et envoyer la valeur de température captée (2.5s – 3.5s). Afin de signaler le bon déroulement de ces opérations, l'application allume un LED pour une demi-seconde (3.5s – 4s). Enfin, l'application passe en mode veille afin de réitérer ce processus après un certain intervalle de temps.

- Pour économiser son énergie, le mote doit souvent se mettre en mode veille. Cela résulte en périodes d'activité de très courtes durées, relativement aux périodes d'inactivité.

Ainsi, un tel profil nécessite un grand niveau de granularité, capable de capter ces fluctuations répétées. Afin de répondre à ces exigences, Jiang et al. [16] ont développé SPOT (Scalable Power Observation Tool), un circuit capable de mesurer la consommation d'énergie d'un mote en offrant une grande précision qui répond au problème des fluctuations. L'inconvénient de cette approche matérielle est qu'elle nécessite un coût additionnel non négligeable, qui est de de 25% par rapport au prix du mote actuel.

Plusieurs approches logicielles ont été développées pour éliminer le coût additionnel de production. Dunkels et al. [8] ont proposé un mécanisme simple se basant sur un modèle linéaire, qui peut être résumé par le formule suivante :

$$\frac{E}{V} = \sum_{i \in C} I_i t_i \tag{1.1}$$

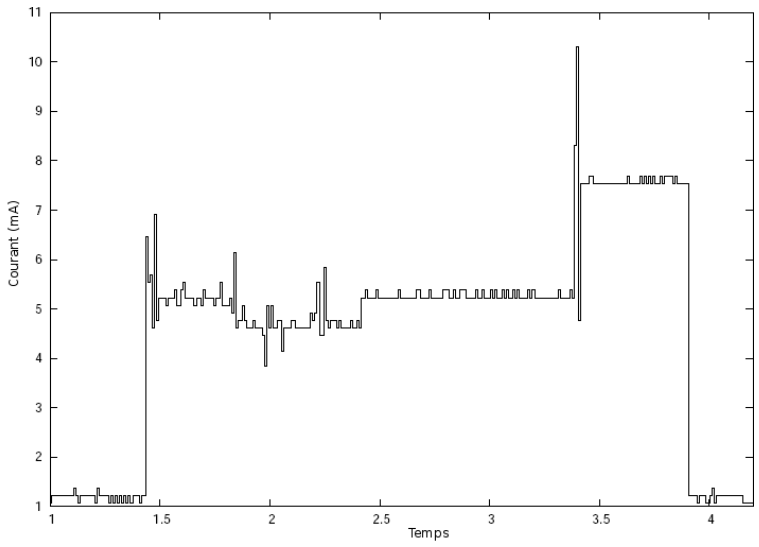


Fig. 1.3: Profil énergétique d'une application WSN typique montrant les différents niveaux de consommation [8].

où :
- E est l'énergie consommée.
- V est le voltage utilisé.
- C est l'ensemble des composants à monitorer, tels que la ratio, l'EEPROM, ... *etc.*
- I_i est le courant nécessaire pour faire fonctionner le composant i.
- t_i est le temps de fonctionnement du composant i.

Pour obtenir les durées de fonctionnement de chaque composant, cette méthode doit modifier légèrement le système d'exploitation utilisé : lorsque ce composant entre en mode de fonctionnement, une estampille est prélevée ; et lorsqu'il est éteint, la différence entre le temps actuel et l'estampille initiale est accumulée par addition dans une variable. Le deuxième paramètre à récupérer est l'intensité du courant nécessaire pour le fonctionnement du composant. Ce paramètre est fixe et est déterminé empiriquement en mesurant le courant utilisé par chaque composant avec des expériences réelles.

1.3.5 La localisation

Dans plusieurs applications, l'information captée est intrinsèquement couplée à la localité où elle a été prélevée. Dans des déploiements de grandes envergures, cette information importante permet une réaction précise et rapide aux différents évènements éventuels. Le système GPS (Global Positionning System) est le mécanisme de localisation le plus commun dans l'industrie actuelle. Toutefois, avec le grand nombre de nœuds déployés, ce système ne peut être employé pour chaque capteur du réseau en raison du coût matériel additionnel. En plus, ce système peut ne pas fonctionner sous certaines conditions extrêmes [26].

Pour y remédier, plusieurs travaux ont été conçus pour répondre aux besoins des WSN [26]. Mais principalement, ces différentes techniques se basent sur deux unités de mesure des signaux : le temps de transfert et la dégradation du signal observée durant ce transfert.

- Connaissant la vitesse de propagation d'un signal donné, le temps de transfert entre deux points a et b permet de déduire la distance ab. Suivant le référentiel temporel considéré, différentes techniques ont été développées telles que : ToA (*Time of Arrival*), TDoA (*Time Difference of Arrival*), Roudtrip Time.
  Néanmoins, les signaux RF émis par l'interface radio des motes ne sont pas adéquats pour ce genre de mesures. Cela est dû à leur vitesse de propagation, qui est trop grande pour être captée par les micro-contrôleurs actuels, qui possèdent des timers de l'ordre des Khz. [2] Pour résoudre ce problème, certaines solutions utilisent des ondes moins rapides, telles que les ultra-sons, qui peuvent être aisément chronométrées à l'aide des timers du mote.

- Au cours de son transfert, un signal subit certaines atténuations affectant son amplitude. En connaissant l'énergie d'émission du signal, ainsi que le modèle de propagation, le récepteur peut estimer la distance parcourue, en fonction de l'énergie du signal reçu. La plus part des interfaces réseaux sans fil actuelles permettent de fournir cette information, connue sous le nom de RSSI (Received Signal Strength Indicator). L'inconvénient de cette méthode est sa faible précision, qui est due à la difficulté de trouver un modèle fiable et réaliste représentant le phénomène de propagation d'une manière fidèle. En effet, ce modèle est intrinsèquement lié à divers paramètres de l'environnement, tels que l'hauteur, les obstacles, la météo, ... *etc*). Cela rend donc difficile l'établissement d'une formule correcte pour une zone d'une grande superficie où ces paramètres peuvent varier d'une façon inattendue.

1.3.6 L'intégration

Pour exploiter pleinement toutes les ressources et avantages d'un WSN, ce dernier doit être fondé sur une architecture ouverte, permettant à diverses applications d'interagir facilement avec l'infrastructure déployée. Une première tentative d'intégration s'est basée principalement sur la donnée. Cette approche, adoptée par TinyDB [34], permet de présenter le réseau de capteurs comme une base de données virtuelle. l'aide d'un langage d'interrogation de données, tel que SQL, des applications externes peuvent interagir avec le réseau, sans connaître pour autant les détails de déploiement ou d'implémentation.

Cependant, un réseau de capteurs n'est pas limité seulement à ses données, mais peut offrir aussi certains services. Par exemple, certaines applications tierces peuvent nécessiter de paramétrer certains nœuds, d'activer des nœuds actionneurs ou de diffuser certaines alertes.

[2] Par exemple, le MSP430 possède un timer de 34Khz, ce qui permet d'avoir une granularité de $2.94117647 \times 10^{-5}$ secondes. Ce niveau de précision n'est pas suffisent pour estimer la durée de transfert d'un signal radio, qui se déplace à la vitesse de la lumière.

Dans ce cas, une intégration par données n'est pas suffisante. En plus, développer une API propriétaire n'est pas une bonne stratégie à moyen et à long terme, vue l'émergence d'architectures hétérogènes nécessitant de communiquer facilement à travers des protocoles ouverts.

Par conséquent, une approche SOA (Service Oriented Architecture) a été introduite dans le domaine des WSN. Dans cette architecture, le réseau de capteurs est vu comme étant un fournisseur de services webs, qui sont définis au niveau de la passerelle logicielle avec le langage WSDL. À l'aide du protocole SOAP, les applications externes peuvent donc facilement interagir avec le réseau, offrant une ouverture totale et contrôlée grâce à une technologie devenue l'approche de facto pour la communication inter-applications hétérogènes.

1.3.7 La sécurité

La sécurité est un maillon essentiel dans toute architecture de communication. Elle vise à protéger le réseau de communication et ses données contre les diverses attaques et comportements malicieux. Comme décrit lors de l'introduction, ce phénomène d'attaque ne cesse d'amplifier à travers le monde, ce qui représente une vraie menace contre tout système informatique.

Suivant la sensibilité du domaine d'application, la sécurité peut couvrir chaque service décrit précédemment. On retrouve dans la littérature divers travaux visant à protéger les différentes pièces de cette architecture complexe, tout en respectant les contraintes physiques et matérielles des WSN. Or, la sécurité est souvent considérée comme une discipline « *gourmande* » en ressources, ce qui a fait apparaître certains challenges intéressants afin d'intégrer ce maillon important dans cet environnement « fragile ».

L'obstacle le plus important reste l'énergie des nœuds. Les motes sont généralement dotés de micro-contrôleurs limités en puissance de calcul et en espace mémoire. Cela rend les opérations cryptographiques assez lentes, ce qui influe négativement sur la consommation en énergie. C'est pour cette raison qu'on remarque une tendance générale dans les solutions WSN qui est l'emploi exclusif de la cryptographie symétrique, à la place des mécanismes à clé publique. Pourtant, les avantages importants de la cryptographie assymétrique sont indéniables, ce qui a poussé les chercheurs à adapter ces mécanismes aux WSN en introduisant la cryptographie elliptique, donnant des résultats assez satisfaisants [31, 24, 36].

L'autre critère important est l'auto-organisation de la solution. Comme les nœuds doivent fonctionner le plus longtemps possible sans intervention humaine, ces solutions doivent être tolérantes aux différentes formes d'attaques et défaillances, sans être dépendantes de paramètres externes. Néanmoins, cette propriété ne doit pas se faire en dépit de la scalabilité de la solution, car le réseau peut comporter un nombre important de nœuds, pouvant aussi varier à travers le temps.

1.4 Exemples d'applications

Le succès d'une technologie est principalement déterminé par son potentiel à offrir des applications utiles et innoventes. C'est ce que les *évangélistes* des WSN ont vite compris, et ont commencé à penser aux éventuelles applications, touchant divers domaines, dès les premières années. Cet effort a débouché effectivement à une adoption croissante de cette technologie par les industriels à travers quelques projets réels, dont on donnera quelques exemples représentatifs à travers les sections suivantes.

1.4.1 Gestion du flux routier

Personne n'ignore les problèmes de congestion et de bouchons routiers présents dans toutes les grandes villes du monde. La plus part de ces villes utilisent des feux de circulations, avérés inefficaces dans plusieurs situations. Cela est expliqué par la nature statique de systèmes utilisés qui ne prend pas en considération les données réelles et actualisées du flux routier. Pour y remédier, la société Sensys Networks propose une solution simple afin d'améliorer la gestion des feux routiers [35]. Afin de connaître le flux actuel des véhicules aux entrées des intersections, la société déploie des détecteurs de vibrations mis sous terre capables de capter le passage d'un véhicule. Avec cette information transmise à un point de calcul via le WSN, ce dernier pourra gérer plus efficacement la coordination entre les feux de plusieurs intersections, facilitant le contrôle du trafic.

Un autre type de solution a été utilisé en Suède avec un succès remarquable. En coopération avec IBM, la ville de Stockholm a lancé un projet unique consistant à taxer automatiquement le trafic dans certaines parties de la ville. En forçant les habitants à payer lors de l'utilisation de certains tronçons routiers, les responsables de la ville espèrent réduire le trafic dans les zones rouges, diminuer la pollution et augmenter l'utilisation du transport en commun. La solution d'IBM consiste à déployer des capteurs capables d'identifier l'immatriculation des voitures, et d'envoyer des factures aux personnes identifiées suivant une certaine tarification. Après le premier mois de mise en service du système, le trafic a été réduit de 25% , les routes ont été soulagées de près de 10.000 véhicules aux heures de pointes, et le transport en commun a connu une augmentation de 40,000 utilisateurs par jour.

1.4.2 Gestion des chaînes logistiques

La gestion des chaînes logistiques (ou SCM : Supply Chain Management) est la discipline regroupant les différents procédés permettant de contrôler et gérer le mouvement d'un produit ou service à partir d'un fournisseur vers un client.

Philips, connaissant l'importance de ce composant au sein de son système de production, s'est associée avec IBM afin d'améliorer sa gestion logistique en réduisant le temps de traite-

ment et les fautes de saisies. La méthode utilisée par Philips avant l'utilisation de ce nouveau système se faisait manuellement en scannant les code-barres collé sur les colis. L'information recueillie est ensuite saisie manuellement afin d'être intégrée à l'ERP de l'entreprise. En plus des erreurs de saisies, les code-barres ne peuvent être lus de n'importe quel angle ou à n'importe quelle distance, infligeant plus de contraintes lors de cette phase de saisie.

En combinant la technologie des WSN et celles des tags RFID, tous ces mécanismes d'identification des colis se font automatiquement sans intervention humaine. Grâce à ce réseau de capteurs, les données de tracking des colis sont automatiquement introduite dans l'ERP, éliminant tout risque de faute de saisie. En plus, les tags RFID utilisent des émissions radio omni-directionnelles, chose qui a éliminé la contrainte de la lecture en LoS (*Line of Sight*) des code-barres.

1.5 Conclusion

À travers ce chapitre, on a présenté un état de l'art survolant les architectures matérielles et logicielles développées pour les WSN. Malgré la courte histoire de cette technologie, un nombre considérable de protocoles et prototypes ont été proposés par la communauté scientifique. Toutefois, cette technologie n'est pas encore arrivée au niveau de maturité requis pour connaître un vrai succès industriel, et certaines questions essentielles restent posées. Cette lacune est principalement visible dans l'aspect logiciel, puisqu'aucune solution n'a été adoptée comme standard par tous les acteurs concernés.

Pour cela, il est important de focaliser cet effort scientifique dans une perspective de normalisation, car cette dernière était le facteur clé du succès de technologies phares, telles que GSM ou Wi-Fi. Vu l'importance de ce point, cet effort de normalisation a récemment été initié par certains pôles industriels, tels que Cisco et ArchRock, en coopération avec l'IETF. Cette normalisation permet de régulariser, et en même temps vulgariser, l'utilisation de cette technologie dans divers domaines de la vie courante.

Chapitre 2

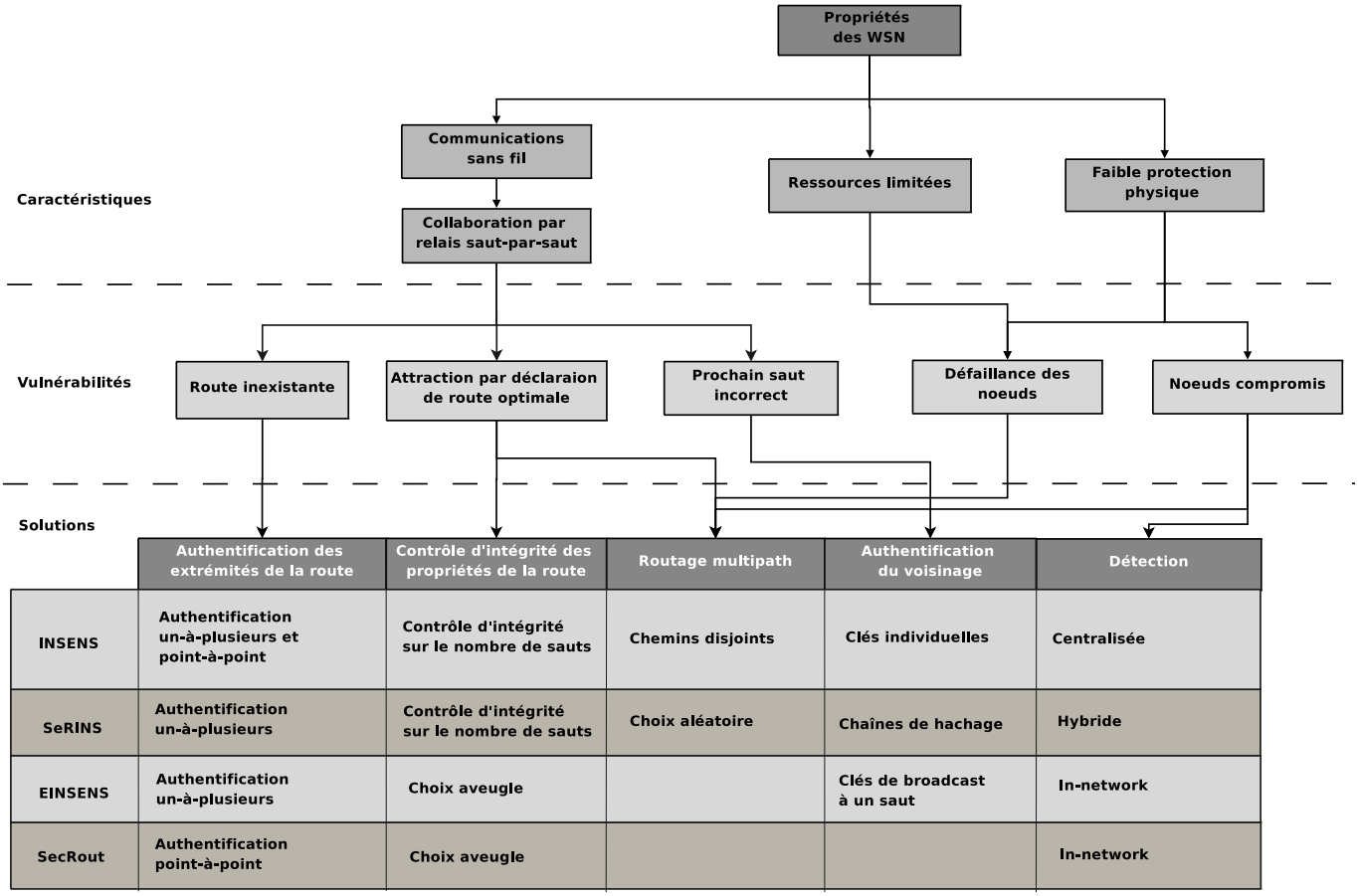
Fiabilité de Dissémination : Problèmes et Solutions

2.1 Introduction

La couche de routage est le module responsable d'acheminer correctement une donnée d'un point du réseau vers un autre [1]. Pour assurer ce rôle, la couche de routage est composée de deux blocs fonctionnels : *la construction des routes* et *le relais des données*. Le premier composant permet de construire et de maintenir une colonne vertébrale logique (ou *backbone*) connectant les nœuds aux destinations désirées via un ensemble de chemins. Le deuxième composant quant à lui utilise cette backbone afin d'acheminer les données captées vers les utilisateurs finaux.

Dans la suite de ce travail, on s'intéressera plus particulièrement au composant de construction des routes qui représente la brique de base du système de communication. Dans ce chapitre, on va présenter une taxonomie détaillée des différents travaux étudiant la fiabilité de la construction des routes dans les WSN. Dans un premier lieu, on va présenter les propriétés des WSN qui sont la source des défaillances dans ce genre de système. Ensuite, on décrira les différentes vulnérabilités découlant de ces propriétés. Cela permettra donc de décrire les objectifs à réaliser et les propriétés à vérifier afin de protéger les WSN contre ces vulnérabilités. Enfin, on terminera par les solutions existantes ayant implémentés ces objectifs et on discutera les inconvénients et avantages de chaque solution. La globalité de cette description est synthétisée par la Fig. 2.1.

[1] A travers ce document, les termes *dissémination* et *routage* désignent le même concept.



	Authentification des extrémités de la route	Contrôle d'intégrité des propriétés de la route	Routage multipath	Authentification du voisinage	Détection
INSENS	Authentification un-à-plusieurs et point-à-point	Contrôle d'intégrité sur le nombre de sauts	Chemins disjoints	Clés individuelles	Centralisée
SeRINS	Authentification un-à-plusieurs	Contrôle d'intégrité sur le nombre de sauts	Choix aléatoire	Chaînes de hachage	Hybride
EINSENS	Authentification un-à-plusieurs	Choix aveugle		Clés de broadcast à un saut	In-network
SecRout	Authentification point-à-point	Choix aveugle			In-network

Fig. 2.1: Taxonomie de la construction sécurisée des routes dans les WSN

2.2 Propriétés des réseaux de capteurs

Les réseaux de capteurs possèdent plusieurs caractéristiques partagées avec les réseaux *ad hoc* traditionnels. Le média sans fil ainsi que le relais saut par saut sont les deux propriétés fondamentales communes entre ces deux familles de réseaux. Néanmoins, les WSN diffèrent sur plusieurs aspects importants. Ces points de différence sont la cause principale de l'inadaptation des solutions MANET aux réseaux de capteurs.

Le premier point de différence est la *faible protection physique*. Contrairement aux systèmes traditionnels, les motes sont dotés de faibles caractéristiques et peuvent être détruits facilement. La raison de cette faible protection est principalement due à la politique de production des motes qui vise à minimiser le coût unitaire en sacrifiant les performances. Ce choix est considéré comme un facteur clé permettant de vulgariser l'utilisation des WSN dans différents domaines de la vie courante. En plus du coût, ces caractéristiques permettent aussi de faire fonctionner les motes sur batterie pour de longues durées, ce qui est d'un grand intérêt pour plusieurs applications des WSN.

Les réseaux de capteurs sont aussi victimes d'un autre critère rendant plus difficile la gestion des pannes. Pour une grande partie d'applications, les capteurs sont déployés dans des zones éloignées ou à accès difficile. En plus, le réseau peut comporter un nombre considérable de capteurs afin d'assurer la couverture d'une grande aire ou bien afin d'améliorer la précision de capture. Ces deux paramètres, qui sont l'*accès difficile* et la *taille du réseau*, rendent le contrôle manuel et individuel des motes très pénible.

2.3 Vulnérabilités des réseaux de capteurs

Les propriétés des réseaux de capteurs sont *à double tranchant*. Certes elles permettent une grande facilité de production et de déploiement, mais rendent le système global de communication assez « *fragile* » à un certain nombre de défaillances. Ces dernières peuvent être classées en deux familles :

2.3.1 Défaillances matérielles

Les caractéristiques des motes existants de nos jours, ainsi que l'environnement de déploiement, rendent les défaillances matérielles très probables. Les causes de ces défaillance peuvent être résumées comme suit :

- Les motes peuvent être cassés ou manipulés par des entités extérieures. En effet, les nœuds déployés en pleine nature attirent des animaux curieux qui peuvent les endommager, ce qui a été effectivement démontré par des déploiements réels dans des champs agricoles où des rongeurs détruisaient les câbles reliant les sondes aux motes [28].
- Le média radio peut subir des perturbations ou des interférences nuisant à la qualité des liaisons. En effet, une étude menée par Zamalloa et Krishnamachari [47] a démontré que les liens radios existants entre les motes actuels sont loin d'être comparables au modèle basique utilisé par les environnements de simulations. En plus, les auteurs démontrent aussi l'existence d'une zone " *transitoire* " pour chaque lien radio, caractérisée par des niveaux de fiabilité assez contrastés.
- Les batteries des nœuds peuvent s'épuiser ; et leur remplacement peut être une tâche assez difficile, vu le nombre important de capteurs et leur emplacement distant dans certains scénarios.

Il est important de remarquer que dans la plus part des cas, ces défaillances sont inévitables, et doivent être considérées comme étant une *propriété intrinsèque* des environnements de réseaux de capteurs, rendant les WSN comme étant des *environnements à faiblesse innée*.

2.3.2 Les attaques

A cause de cette faiblesse inévitable, les WSN représentent aussi une proie facile pour les intrus. Comme pour tout autre système de communication, la sécurité doit être considérée comme un bloc fonctionnel clé de l'architecture, plus spécialement pour les systèmes sans fil. Il faudra alors définir clairement les profils possibles d'intrus, ainsi que les buts visés par les différentes attaques.

Un WSN peut être attaqué par deux types d'intrus :

a. *Les intrus internes*
   La faible protection physique permet à un attaquant de prendre le contrôle d'un nœud légitime du réseau. Cela signifie que l'intrus aura accès à toutes les données secrètes de l'application, faisant apparaître ce nouveau nœud malicieux comme étant un nœud authentifié du réseau.
   Pour y parvenir, l'attaquant peut employer la méthode manuelle en reprogrammant le mote à travers son port série par exemple. Si le réseau dispose d'un mécanisme de reprogrammation à distance, l'intrus peut essayer d'attaquer ce mécanisme afin de propager son propre code dans tous le réseau ! Heureusement, certaines solutions ont été développées afin de protéger ce mécanisme de mise à jour et authentifier toute nouvelle requête de reprogrammation [39].
   Une troisième méthode assez particulière, et peu exploitée, se base sur l'exploitation de failles logicielles donnant accès aux systèmes distants, telles que les débordements de buffer (*buffer overflows*). En exploitant certaines erreurs de programmation, un intrus peut injecter certaines données corrompues afin de contrôler le flux d'exécution du système cible. Cette attaque est l'une des attaques les plus dangereuses dans les systèmes informatiques, ce qui a débouché à de nombreuses protections dans les systèmes traditionnels [11]. Néanmoins, les WSN ne bénéficient pas encore de protections valables, et cette piste reste pleine de questions en attente [4].
b. *Les intrus externes*
   Un intrus externe est un nœud étranger du réseau ne faisant pas partie des nœuds déployés par l'administrateur. Un intrus externe ne peut pas avoir accès aux informations sauvegardées par les membres légitimes du réseau (tels que les clés de chiffrement, les secrets partagés, ... *etc*). Cependant, certains intrus externes peuvent avoir des caractéristiques supérieures à celles des motes déployés, leur permettant d'amplifier l'impact de leurs attaques. Par exemple, un intrus peut utiliser un ordinateur portable afin d'avoir une plus grande puissance de calcul ou une plus grande portée de communication.

Ces différents profils d'intrus peuvent mener diverses attaques au niveau de la couche de routage. Plusieurs travaux ont été publiés décrivant les différentes formes d'attaques contre le routage dans les WSN [18]. Certaines de ces attaques sont héritées des réseaux ad hoc traditionnels, tandis que d'autres sont propres aux WSN. Dans le reste de ce paragraphe, on présentera une explication intuitive des attaques possibles afin de bien cerner les vulnérabilités à combattre :

Le routage multi-saut est basé sur un *effort collectif* visant à établir la topologie de communication et d'en faire usage lors du relais des données. Cette forme de collaboration représente en même temps le cœur des réseaux multi-sauts, mais aussi la *source primaire des failles* de sécurité au niveau de la couche routage. En effet, une collaboration dans un environnement distribué nécessite un comportement *rigoureusement orchestré* suivant des règles préétablies. Malheureusement, un intrus peut enfreindre ces règles en perpétrant ingénieusement une suite d'actions qui permettent de dévier le fonctionnement normal du système vers un comportement voulu par l'intrus. Cela peut se résumer à deux objectifs :

- Un attaquant peut essayer d'*attirer le maximum de routes* passant par lui afin d'octroyer une place prédominante au sein de la topologie de routage. Cette position permettra par la suite de contrôler le maximum de flux de données traversant l'entourage de l'intrus. D'une manière générale, ce type d'attaques est menée en injectant de *fausses déclarations de routes optimales*. Cela signifie que l'attaquant doit « mentir » en déclarant qu'il possède la route la plus optimale vers une destination donnée. Sans mécanisme de sécurité, ces déclarations vont placer l'attaquant en position de force parmi les routeurs du réseau. Dans un cas extrême, l'attaquant peut même prétendre d'être la SB, collectant ainsi *tout* le trafic du réseau !

- La deuxième catégorie d'attaques vise à *inhiber la fonction de routage* dans les nœuds cibles afin de les déconnecter du reste du réseau. Cela peut prendre deux formes : *logique* et *physique*.

  L'*inhibition logique* consiste à « *empoisonner* » les nœuds par des informations de routage inconsistantes. Cela va fausser la vue locale de routage des nœuds victimes, engendrant une déconnexion logique vu que les routes découvertes sont inutilisables. Les boucles de routage ainsi que les fausses déclarations de voisinage sont deux exemples typiques de ce genre d'attaques.

  L'*inhibition physique* quant à elle essaye de profiter des contraintes des motes afin de les déconnecter du reste du réseau. Parmi ces attaques de déni de services (Denial Of Service), on peut citer l'attaque d'exhaustion d'énergie qui vise à forcer les nœuds à effectuer des calculs ou des communications inutiles. Cela gagne en envergure lorsque le protocole cible emploie des primitives cryptographies durant la découverte des routes.[2]

2.4 Mécanismes de défense

Dans le but de se protéger contre ces diverses vulnérabilités, il est impératif de renforcer les mécanismes de collaborations avec des primitives de sécurité permettant de minimiser le plus possible l'impact de ces problèmes. En plus, il faut considérer le problème des pannes naturelles, puisqu'elles représentent une propriété normale des WSN. Cette section présente

[2]L'inhibition physique peut aussi être menée au niveau physique en détruisant le capteur par exemple. Mais dans ce travail, on s'intéresse plus particulièrement aux vulnérabilités au niveau de la couche de routage.

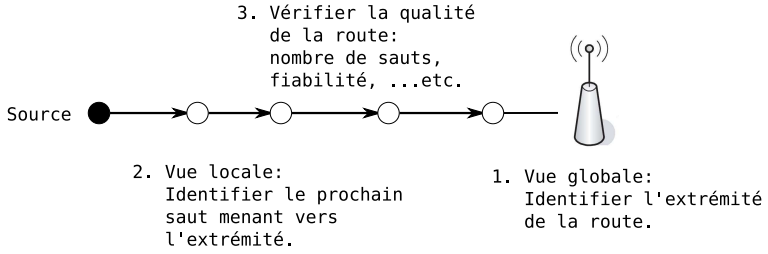


Fig. 2.2: Les informations à protéger pour identifier une route valide

les différentes solutions développées pour sécuriser la construction des routes, dont les plus représentatives sont : INSENS [5], EINSENS [6], SeRINS [22] et SecRout [46]. Ces solutions proposées s'articulent autour de trois axes majeurs : la *prévention*, la *détection* et la *tolérance*.

2.4.1 La prévention

La prévention consiste à écarter le risque d'attaque en implémentant un ensemble de mécanismes de protection contre la manipulation illicite des informations de routage. Néanmoins, ces informations possèdent des rôles et des natures différentes, nécessitant chacune un mécanisme de protection adéquat. On peut distinguer entre trois catégories d'informations de routage (voir Fig. 2.2) :

- Les identités des extrémités (source, destination) représentent la propriété principale de la route. C'est la vue point-à-point globale du nœud sur le chemin construit.
- L'identité des nœuds relais constitue la vue locale du chemin, et elle est généralement réduite à l'information du prochain saut seulement (*i.e.* l'information du voisinage menant à l'extrémité de la route).
- Une route peut aussi être caractérisée par certaines informations qualitatives. Ces propriétés sont utilisées en tant que filtre afin de permettre le choix entre différentes alternatives. Certaines informations sont de nature *mutable*, signifiant que les nœuds ont le droit de les modifier durant le relais. La longueur de la route est un exemple typique des champs mutables permettant de construire des chemins courts tout en garantissant l'absence de boucles de routage. D'autres informations, telles que l'estampille (indicateur de fraîcheur), sont non mutables et doivent être préservées intactes durant toute la propagation de la requête.

Chacune de ces informations nécessite un mécanisme de protection propre à elle. Dans ce qui suit, on présentera le type de protection à fournir pour chaque catégorie d'information ainsi que les solutions existantes implémentant ces mécanismes.

2.4.1.1 Protection de l'identité des extrémités de la route

L'authentification est le mécanisme le plus adéquat pour la protection des identités. En identifiant les extrémités de la route, on garantit l'existence du chemin vers l'entité authentifiée, sans pour autant garantir sa validité (ce qui se fait avec les deux protections restantes). Cette authentification peut s'établir sous deux formes :

Dans les topologies classiques des réseaux ad hoc, la construction des routes se fait entre une source (un capteur) et une destination (la SB) suivant le schéma traditionnel RREQ-RREP. Comme la communication est un-à-un, une clé secrète par capteur suffit afin d'établir une authentification entre les deux extrémités.

Dans les topologies arbre, la construction se fait suivant le paradigme un-à-plusieurs où la SB diffuse une RREQ vers tout le réseau afin de construire des chemins *unidirectionnels* vers la SB. Donc, les motes ont besoin d'authentifier seulement la SB afin de garantir l'existence de la route. Néanmoins, cette authentification un-à-plusieurs ne peut se faire avec de simples clés secrètes symétriques et nécessite une certaine propriété *asymétrique*. Pour cela, toutes les solutions existantes utilisent des *chaînes de hachage à sens unique*. Comme ce concept sera souvent évoqué durant ce travail, on va présenter une brève description de ce mécanisme :

Une chaîne de hachage à sens unique (ou OHC : *One-way Hash Chain*) [20] est utilisée comme générateur de numéros de séquence à sens unique (OWS : *One-Way Sequence number*), ce qui fournit un moyen efficace pour l'authentification d'une séquence de messages dans une communication un-à-plusieurs (voir Fig. 2.3). Elle consiste en une séquence de nombres $(K_i)_{0 \leq i \leq n}$ générées à partir d'une graine aléatoire K_n comme suit :

$$K_i = F\left(K_{i+1}\right), 0 \leq i < n \tag{2.1}$$

où F est une fonction à sens unique. Cette chaîne entière est sauvegardée au niveau du nœud source de la communication, alors que les éventuels récepteurs sont chargés avec la première valeur K_0 de la chaîne (appelée graine d'authentification), qui va servir comme vérificateur V. Pour chaque message émis, le nœud source révèle une nouvelle valeur de la chaîne dans l'ordre inverse de génération, *i.e.* K_1 puis K_2 et ainsi de suite. Comme les nœuds ont initialisé leur vérificateur V avec la première valeur de la chaîne K_0, ils peuvent vérifier l'appartenance de n'importe quelle valeur reçue v à la chaîne en question comme suit :

$$\exists n : V = F^n\left(v\right) \tag{2.2}$$

Si la valeur v est authentifiée, le vérificateur V est mis à jour avec cette valeur pour authentifier les prochains éléments de la chaîne.

Dans les protocoles INSENS, EINSENS et SeRINS, ce mécanisme est employé comme suit : la SB étant l'initiateur de la construction, elle génère une OHC avant le déploiement des nœuds. Lorsqu'un nœud est rajouté au réseau, il est chargé (manuellement) avec la première valeur non divulguée de la chaîne (la valeur en cours de la chaîne). A chaque reconstruction

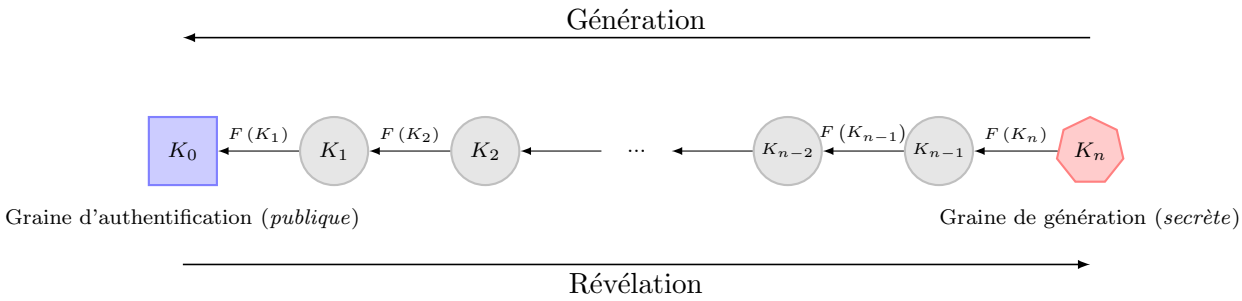


Fig. 2.3: Concept de *chaînes de hachage à sens unique*. Initialement, seule la valeur K_0 est publique. Lorsque la source veut authentifier son premier message, elle divulgue la valeur K_1. Comme $K_0 = F(K_1)$ et que F est à sens unique, chaque récepteur peut vérifier l'appartenance de K_1 à la chaîne de K_0, sans pouvoir deviner K_1 à l'avance.

de la topologie, la SB divulgue la prochaine valeur de la chaîne dans le message RREQ émis. Lors du relais de ce message, chaque nœud du réseau peut authentifier la valeur émise grâce au vérificateur de la chaîne, et aucun nœud ne peut deviner à priori la prochaine valeur d'authentification.

Il faut noter qu'en plus de l'authentification de l'extrémité, le mécanisme doit assurer une propriété de dynamisme permettant d'éviter les attaques de re-jeu. En effet, les topologies de routage sont généralement reconstruites durant des intervalles réguliers appelés *tours*. Il faudra donc garantir que les preuves d'authentification utilisées pendant un tour i seront invalides pour tout tour $j > i$. Les chaînes de hachages possèdent cette propriété de dynamisme d'une manière innée, vu qu'une nouvelle valeur est divulguée à chaque nouvelle authentification et que ces chaînes sont à sens unique. Cela signifie que les chaînes de hachage peuvent être utilisées pour deux objectifs : l'authentification de la source et l'identification du tour. Par contre, l'authentification par clé partagée ne possèdent pas cette propriété, et nécessitent l'utilisation d'artifices supplémentaires tels que des numéros de séquences ou les nonces.

2.4.1.2 Protection de l'identité des nœuds relais

L'authentification des nœuds relais est nécessaire afin d'assurer la validité des identificateurs des prochains sauts. Cette défense permet de contrer les tentatives de fausses déclarations de voisinage, telles que le Hello Flooding [18] et l'usurpation d'identité. Pour éviter ces attaques, chaque capteur doit découvrir et authentifier chaque voisin direct lié par un lien bi-directionnel en utilisant des mécanismes défi-réponse. Cet ensemble de voisins est nommé *voisinage atteignable*. Les protocoles existants proposent trois méthodes pour l'authentification des voisins atteignables :

– Dans INSENS, les capteurs sont chargés par des clés individuelles secrètes partagées avec la SB. Chaque nœud calcule un secret avec sa propre clé et le communique à ses voisins. Ces derniers vont utiliser ce secret comme preuve de voisinage qui va être transmise à

la SB. La SB va comparer les preuves reçues des présumés voisins et peut donc déceler les relations inconsistantes, vu que la SB peut régénérer le secret de chaque nœud du réseau.

- L'approche adoptée par EINSENS se résume à l'utilisation de clés de broadcast à un saut. Ces clés sont générées par chaque nœud et sont partagées avec les voisins directs en utilisant des mécanismes tels que LEAP [48, 49]. Chaque message de contrôle émis doit être authentifié à l'aide de cette clé. Mais cette approche n'est pas protégée totalement contre les attaques d'usurpation d'identité car un nœud connaissant la clé de broadcast de chaque voisin peut générer sa preuve d'authentification. Cette lacune est due au fait que les clés partagées secrètes ne sont pas adéquates aux communications de broadcast, vu la multiplicité des récepteurs et l'unicité du secret partagé.
- L'approche de SeRINS permet de pallier le problème précédent en rajoutant l'asymétrie manquante. Au lieu de partager un secret avec les voisins directs, chaque nœud génère une chaîne de hachage et divulgue le premier élément aux voisins directs atteignables [21]. Chaque message futur comportera la prochaine valeur de la chaîne qui permettra d'authentifier d'une manière plus confidente la source du message.

2.4.1.3 Protection des propriétés de la route

Mis à part les différents types d'identités à protéger, les autres propriétés de la route sont généralement optionnelles, car elles ne sont utilisées que pour « *améliorer* » les performances offertes par la topologie résultante.

Pour cette raison, le protocole EINSENS a choisi la solution « facile » en effectuant un *choix aveugle*. En effet, le protocole n'assure que la validité des identités de nœuds participants à la découverte des routes, tandis que le choix des chemins se fait d'une manière FIFO : première route valide découverte, première choisie. Ce genre de politique permet de limiter le champs d'action de l'attaquant vu que les routes ne possèdent pas de propriétés qui font que le choix des nœuds soit déterministe. L'absence de déterminisme est effectivement une bonne protection car elle empêche l'attaquant de contrôler le choix de route de ses voisins.

Dans le but d'améliorer les performances de routage, les protocoles INSENS et SeRINS ont adopté la métrique du nombre de sauts comme critère de choix des routes. Cette métrique étant mutable, tout nœud a la possibilité de la modifier, rendant sa protection assez délicate :

- Pour cela, INSENS a choisi une approche centralisée où toutes les vérifications sont effectuées au niveau de la SB. Cette dernière collecte les informations de voisinage de chaque nœud du réseau, vérifie les preuves de voisinage et génère un graphe local représentant toutes la topologie physique de communication. Ce graphe sera utilisé pour construire les chemins de routage les plus courts pour chaque nœud du réseau.
- L'approche centralisée permet d'éviter n'importe quel comportement malicieux, mais souffre d'une très faible scalabilité vu le nombre important de messages de contrôle.

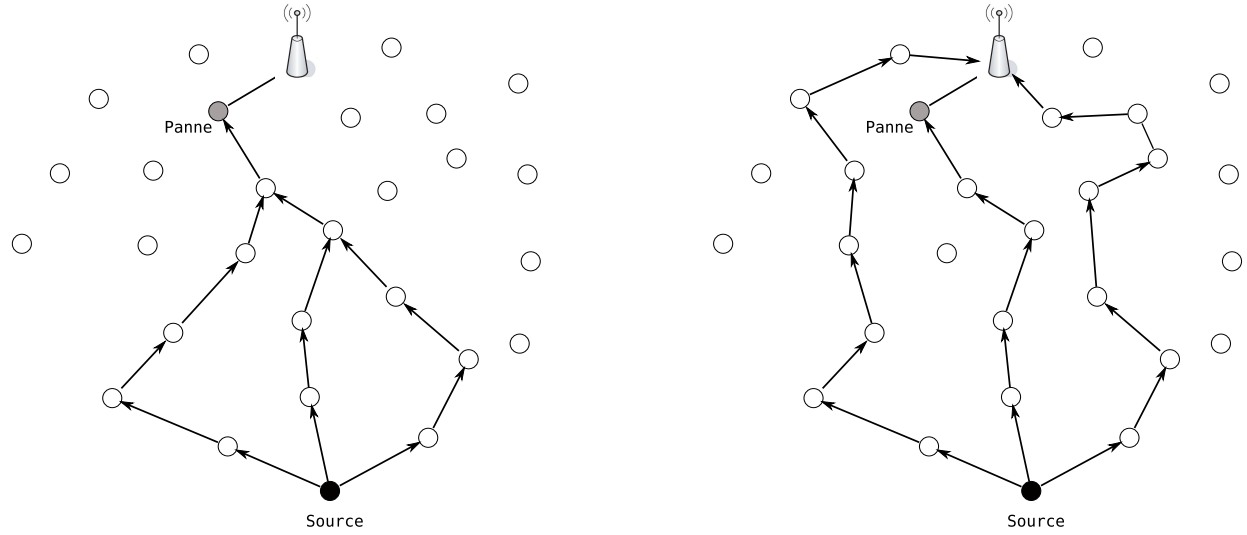
Pour cela, SeRINS adopte une vérification semi-distribuée qui prend avantage du fait que le nombre de sauts n'est pas une information totalement mutable. En effet, un nœud n'a le droit d'incrémenter cette valeur que d'une seule unité à chaque saut. Pour vérifier ce comportement, SeRINS utilise des chaînes de hachage construite « *à la volée* » durant le relais. Contrairement aux chaînes classiques, qui sont générées complètement et sauvegardées au niveau de la SB, cette dernière communique la graine de génération r de la chaîne dans le message RREQ initial. À chaque relais du message, le nœud génère la prochaine valeur de la chaîne en utilisant la valeur reçue. Ainsi, à une distance d, la valeur de la chaîne doit être $F^d(r)$, créant ainsi un mapping entre nombre de sauts et les valeurs de la chaîne. Ce mapping permet de vérifier si deux RREQ réfèrent à deux distances « cohérentes ». En recevant deux RREQ $\langle d_1, v_1 \rangle$ (où d_1 est la distance en nombre de sauts et v_1 est la valeur de hachage correspondante) et $\langle d_2, v_2 \rangle$, le nœud peut vérifier si elles sont cohérentes en vérifiant que :

$$\begin{cases} v_1 = F^{d_1 - d_2}(v_2), \text{si } d_1 > d_2 \\ v_2 = F^{d_2 - d_1}(v_1), \text{sinon} \end{cases} \tag{2.3}$$

2.4.2 La détection

La détection représente le mécanisme central de chaque protocole de sécurité. Il définit la procédure curative lors de la violation de l'une des règles établies par le mécanisme de protection décrit précédemment. Trois approches différentes ont été développées offrant trois niveaux de scalabilité :

- *Les solutions centralisées*, telles que INSENS, effectuent toutes les vérifications de sécurité au niveau de la SB. Comme décrit dans le paragraphe précédent, la SB est responsable de trouver les fausses déclarations de voisinage et de créer les routes « valides ». Ainsi, les capteurs sont démunis de toute tâche de vérification ou de choix de chemins. En plus, la SB possède une maîtrise totale sur la topologie de routage et peut créer n'importe quel type de chemins.
- *L'approche distribuée* adoptée par EINSENS est basée sur une vérification in-network effectuée uniquement par les capteurs sans aucune aide de la SB. Les opérations de sécurité sont effectuées saut par saut durant le relais des messages RREQ. Dans ce genre d'approches, les motes utilisent leur information locale pour détecter les comportements malicieux, sans avoir à communiquer avec leurs voisins ou bien la SB. Par conséquent, la détection des attaques est instantanée et n'engendre aucun délai ou message supplémentaire.
- *Une approche hybride* intermédiaire développée par SeRINS consiste à déléguer partiellement la tâche de vérification aux motes. A cause de cette information partielle, les capteurs ne peuvent pas prendre des décisions instantanées lors de la détection d'une incohérence. Ils doivent alerter la SB qui interrogera plus de nœuds afin de recueillir l'information manquante et prendre la décisions adéquate.



(a) Multi-chemins avec intersection (b) Multi-chemins disjoints

Fig. 2.4: La figure 2.4(a) montre l'impact d'une panne sur une topologie multi-chemins qui ne considère pas l'intersection des routes, comme pour SeRINS. On remarque qu'une seule panne suffit pour déconnecter le nœud source de la SB. Par contre, les chemins disjoints, représentés par la Fig. 2.4(b), résistent mieux à l'occurrence de la même panne et dans la même topologie.

2.4.3 La tolérance

Comme les capteurs peuvent être compromis ou tomber en panne, les routes construites peuvent devenir inutilisables. La solution triviale reste la re-découverte périodique de nouvelles routes, ce qui engendre un coût supplémentaire en énergie. Afin d'éviter les retombées négatives de cette reconstruction périodique parfois inutile, les motes peuvent « *anticiper* » la défaillance des nœuds en prévoyant des routes *alternatives*. Les routes alternatives sont un moyen d'assurer une certaine *redondance* lors du transport de la donnée. Cette redondance est nécessaire pour garantir la continuité du relais lors de la présence de déconnexions de motes et de liens.

La redondance offerte par les routes alternatives est caractérisée principalement par le *taux de duplication* des données relayées, qui est un paramètre crucial dans la définition du taux de tolérance. Dans un contexte de routage, cette duplication recherchée est étroitement lié au taux d'intersection des routes alternatives. Ce concept est expliqué par les deux points suivants :

– Dans SeRINS, les chemins les plus courts sont choisis comme chemins alternatifs. Cette approche n'est pas très intéressante en terme de tolérance car elle ne prend pas en considération l'intersection des chemins. En effet, si les chemins alternatifs d'un nœud se croisent, une panne peut « *couper* » plusieurs chemins, rendant la topologie assez fragile. Comme le montre la Fig. 2.4(a), une panne suffit pour déconnecter le nœud source de la SB.

– Pour limiter l'impact des pannes sur la connexité de chaque nœud, INSENS choisit

des chemins totalement disjoints pour chaque nœud. En évitant l'intersection entre les chemins, une panne n'infectera qu'un seul chemin au maximum pour chaque nœud du réseau (comme le montre la Fig. 2.4(b)).

La Fig. 2.4 montre aussi que le nombre de chemins découverts n'est pas une métrique décisive pour comparer la tolérance des topologies, puisque les deux scénarios décrits par la figure présentent le même nombre de chemins alternatifs mais avec une tolérance largement différente.

2.5 Conclusion

Les routes multi-sauts représentent la colonne vertébrale de communication pour les WSN. Il est impératif de bâtir une colonne vertébrale solide pour ce genre de systèmes caractérisés par une faiblesse innée. Cette backbone doit pouvoir connecter les nœuds du réseau avec la SB en considérant les différents types de défaillances possibles. Ainsi, une conception d'une couche de routage *tolérante* reste un facteur clé pour améliorer la durée de vie du réseau.

Ce chapitre a présenté une synthèse générale des solutions proposées dans ce contexte, ce qui permet d'avoir une vue précise englobant touts les problèmes rencontrés lors de la conception d'un nouveau protocole fiable de dissémination. Pour étendre cette vue à un niveau plus élémentaire, le prochain chapitre donnera tous les détails de fonctionnement de chaque protocole étudié précédemment.

Chapitre 3

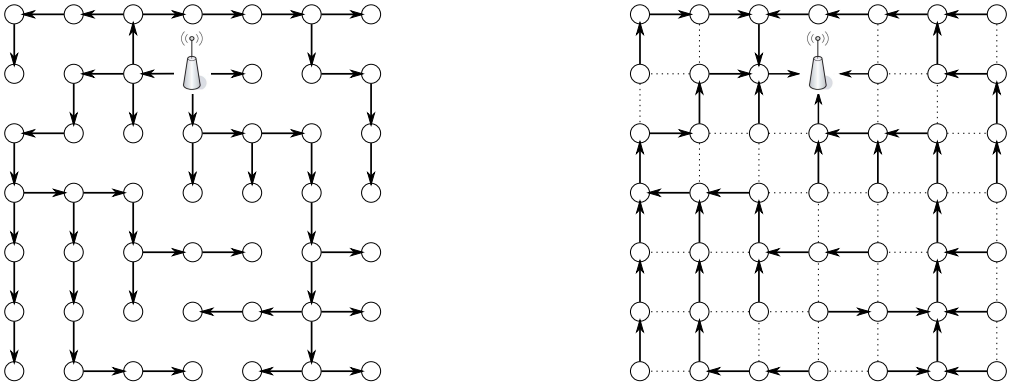
Les Protocoles de Fiabilité du Routage

Dans ce chapitre, on présentera les détails de fonctionnement des protocoles INSENS, EINSENS, SeRINS et SecRout. Pour cela, on utilisera les notations suivantes :

- $E(K, D)$: l'information D est cryptée avec la clé secrète K.
- $MAC(K, D)$: code d'authentification sur la donnée D en utilisant la clé K.
- $A \rightarrow B : X$: l'entité A émet le message X vers B.
- $A \rightarrow * : X$: l'entité A émet le message X à ses voisins directs.
- N_A : un nonce généré par A.
- $X \mid Y$: X concaténé à Y.
- F : une fonction à sens unique.
- ows : un numéro de séquence à sense unique afin d'identifier le tour et authentifier la SB. Il est prélevé d'une chaîne de hachage stockée au niveau de la SB. Chaque capteur est initialisé avec un vérificateur V_{ows}, représentant la première valeur non divulguée de la chaîne. V_{ows} sera mis à jour à chaque réception d'une nouvelle valeur valide appartenant à la chaîne.
- K_i : clé secrète de i, connue seulement par i et la SB.
- $K_{i,j}$: clé secrète partagée entre i et j.
- BK_i : clé de broadcast de i, générée par i et partagée avec ses voisins directs.

3.1 INSENS

INSENS (*INtrusion-tolerant routing for wireless SEnsor NetworkS*) [5] est l'un des premiers protocoles de routage sécurisé pour les WSN. Il vise principalement à centraliser la construction des routes au niveau de la SB. Cela permet à la SB de tracer une *cartographie correcte* du réseau qui permettra d'établir les tables de routage pour chaque capteur. Ces tables seront



(a) L'arbre temporaire (b) Collection des états de lien

FIG. 3.1: Phase de collecte des états de liens. (a) La première étape consiste à établir un arbre temporaire afin de permettre une communication multi-sauts pour la collecte des états de lien lors de la prochaine étape. (b) La seconde étape débute par la collection des états de liens (schématisés avec les lignes en pointillés). Ces preuves seront transmises via l'arbre temporaire.

transmises par la suite aux nœuds concernés de façon sécurisée.

Le protocole se compose de deux phases : *la collecte des états de liens* et *la construction des tables de routage.*

3.1.1 Collecte des états de liens

La SB déclenche périodiquement une nouvelle construction des routes en collectant les informations de lien de chaque capteur du réseau. Ces informations représentent la liste des voisins, accompagnée de preuves permettant de vérifier la cohérence des déclarations de voisinage. Comme les nœuds distants doivent envoyer cette information à la SB via des chemins, et que les chemins ne peuvent être construits qu'après la collecte de ces informations, INSENS établit un arbre temporaire permettent d'acheminer les états de lien vers la SB. La construction de cet arbre est basée sur une simple diffusion générale d'un message RREQ initié par la SB et ayant le format suivant :

$$i \rightarrow * : i, ows, MAC(K_i, i \mid\mid ows) \tag{3.1}$$

La valeur $MAC(K_i, i||ows)$ est fournie comme *preuve de voisinage* destinée aux nœuds dans la portée de i. Comme cette preuve est calculée à l'aide de la clé secrète K_i sur une donnée commune et rafraîchie ows, un intrus ne pourra pas fabriquer une telle valeur s'il n'est pas voisin de i.

Pour établir l'arbre temporaire, chaque capteur choisit l'émetteur de la première RREQ reçue dans un tour comme son parent. Après ce choix, le nœud relaie le message RREQ en mettant à jour le champs de l'identificateur de l'émetteur et la preuve de voisinage. Comme

les voisins du nœud relais vont effectuer les mêmes opérations, chaque nœud peut collecter les preuves de ses voisins.

Ces informations d'état de lien collectées ls_i, constituées des identificateurs des voisins découverts ainsi que leurs preuves respectives, sont envoyées vers la SB en utilisant le message RREP suivant :

$$i \rightarrow * : i, ls_i, MAC(K_i, ls_i \mid\mid ows), MAC_p \tag{3.2}$$

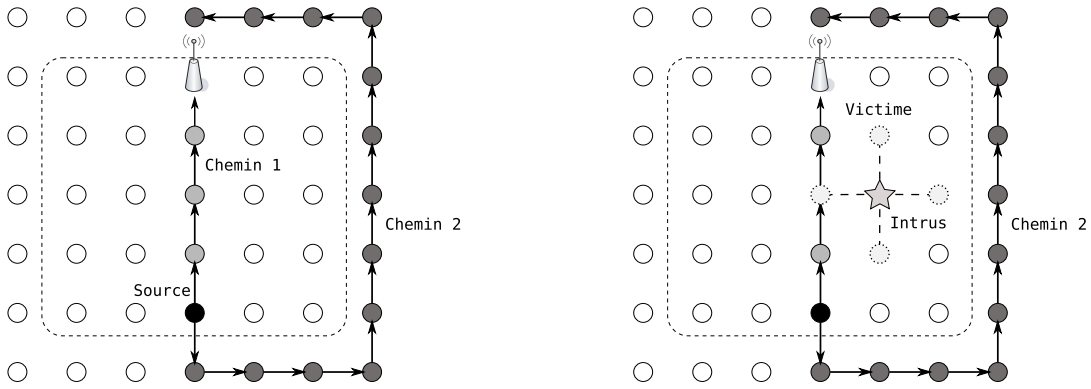
Le champ MAC_p est la preuve de voisinage du parent sélectionné. Ce message RREP est relayé de parent en parent au sein de l'arbre temporaire jusqu'à arriver à la SB. Notons que ce message est envoyé en broadcast local sans spécifier l'adresse du parent afin de la cacher. Le parent pourra détecter ce message à l'aide de la valeur MAC_p, .

3.1.2 Construction des tables de routage

Après réception des messages RREP, la SB débute la construction d'une cartographie correcte reflétant les liens physiques *réels* rapportés par la phase précédente. Pour déceler les fausses déclarations de voisinage, la SB regénère les preuves de chaque capteur, et compare ces valeurs avec celles rapportées par les « *prétendus* » voisins.

Cette phase de filtrage permet d'obtenir un graphe de connexité englobant tout le réseau. Pour obtenir les chemins de chaque nœud, la SB peut appliquer n'importe quel algorithme de recherche de chemins, tels que Dijkstra. Dans un but de renforcer la tolérance de la topologie, les auteurs d'INSENS ont choisi de construire pour chaque nœud deux chemins disjoints, mais distants au minimum de deux sauts. Cela signifie que chaque nœud du premier chemin doit être situé à une distance supérieure à deux sauts de tout autre nœud du second chemin. Ce concept et son utilité sont illustrés avec la Fig. 3.2.

La découverte de l'ensemble des chemins permettra d'établir la table de routage de chaque nœud. La table de routage d'un nœud i contient les identificateurs de la source et du prochain saut pour chaque chemin passant par i. Afin de faire parvenir cette table au nœud i, la SB doit découvrir un chemin approprié, car l'arbre temporaire établi lors de la première phase ne contient que des chemins unidirectionnels partant des nœuds vers la SB. Pour éviter d'encombrer le réseau avec une nouvelle découverte de chemins, la SB opère par *anneaux* en envoyant ces tables aux nœuds se trouvant à h sauts avant ceux à $h + 1$ sauts. Quand un nœud reçoit une table envoyée à un autre nœud, il doit vérifier si le ce nœud destinataire est inclus dans sa table de routage comme source d'un des chemins passant par lui. Dans ce cas, le nœud récepteur relaie simplement ce message vers ses voisins, qui feront de même jusqu'à arriver à destination.



(a) Sélection des deux chemins

(b) Avantage de la distance minimale entre les deux chemins

FIG. 3.2: Construction des routes avec le protocole INSENS. (a) Le chemin 1 est obtenu avec un algithme du plus court chemin, tel que celui de Dijkstra. Par la suite, tous les nœuds appartenant au voisinage à deux sauts des membres du chemin 1 sont écartés du graphe, et le deuxième chemin est obtenu en appliquant le même algorithme sur le nouveau graphe. (b) L'intêret de cette distance minimale de deux sauts apparaît lors d'une attaque. Lorsqu'un intrus attaque une région, il ne peut *contaminer* qu'un seul chemin au maximum. Ainsi, INSENS garantit que n'importe quel nœud du réseau aura un chemin actif à la suite d'une attaque.

3.1.3 Analyse

La centralisation de la construction au niveau de la SB permet d'établir une vue globale sur la topologie de communication du réseau entier. Cela offre un contrôle total de la qualité des routes construites, mais ce mécanisme souffre d'une très faible scalabilité. Le nombre de messages nécessaires pour collecter les informations de liens accroît d'une manière exponentielle à cause du relais unicast et individuel de chaque message d'état du lien vers la SB. Comme chaque nœud est responsable du relais de tous les messages des membres de son sous-arbre, ce processus de collecte engendrera une grande consommation d'énergie et un considérable temps de convergence, avec une grande probabilité de collision.

Pour illustrer l'ampleur de cet impact, considérons un réseau organisé en arbre binaire de profondeur n. Sachant qu'un nœud à un niveau i nécessite i relais pour atteindre la SB, et que le niveau i contient 2^i nœuds, on peut facilement déterminer le nombre total des messages relayés :

$$\sum_{i=1}^{n} i2^i = 2 + (n-1)2^{n+1} \tag{3.3}$$

La figure 3.3 montre la variation de cette grandeur par rapport à la profondeur n de l'arbre. On remarque une croissance exponentielle, augmentant très rapidement ce qui est un signe clair de non scalabilité. Par exemple, avec un arbre de 10 sauts, les motes doivent s'échanger 18 434 messages, ce qui est un grand nombre vu que le réseau ne contient que 1 024 nœuds. En plus, la charge de relais n'est pas uniforme dans tout le réseau, puisque chaque nœud est

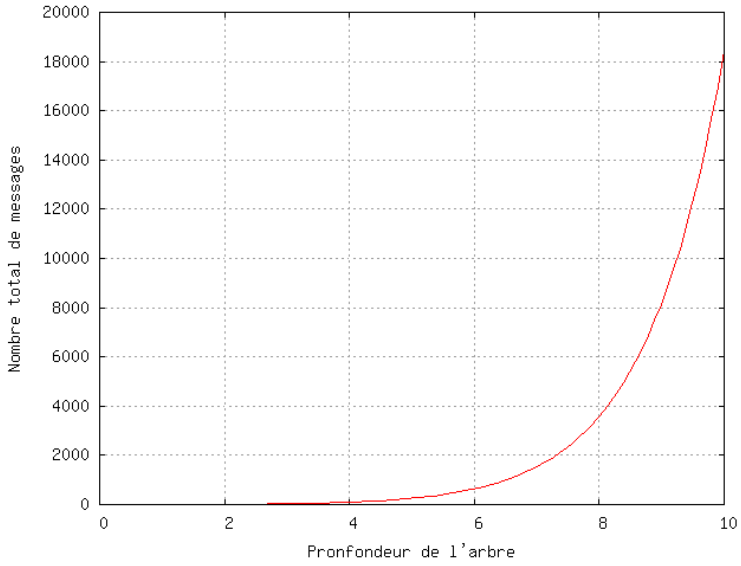


FIG. 3.3: Nombre de messages échangés lors de la collecte des états de lien

responsable de transmettre les messages de tous les membre de son sous-arbre. Par conséquent, plus on se rapproche de la SB, plus les nœuds devront transmettre plus de messages.

3.2 EINSENS

EINSENS (*Enhanced INtrusion-tolerant routing for wireless SEnsor NetworkS*) [6] a été proposé par les mêmes auteurs d'INSENS afin d'offrir une meilleure scalabilité que l'approche centralisée. Toutefois, cette amélioration a causé une dégradation de la tolérance de la topologie, vu que la nouvelle version ne fournit *qu'un seul chemin* pour chaque nœud. Pour palier à ce problème, les auteurs proposent d'*émuler* une topologie multi-chemins en déployant plusieurs SB et de construire, pour chaque nœud, un chemin vers chacune des SB (voir Fig. 3.4).

La nature distribuée d'EINSENS impose aussi une autre contrainte. Afin de permettre une vérification in-network, les motes voisins doivent établir des clés secrètes pour pouvoir s'échanger ultérieurement des preuves vérifiables. L'établissement de ces clés nécessite néanmoins plus de messages de contrôle et plus d'espace de stockage, ce qui n'était pas nécessaire dans INSENS puisque la SB effectue toutes les opérations cryptographiques de vérification.

Le protocole EINSENS est donc un protocole sécurisé uni-chemin pouvant être considéré comme étant la version sécurisée du protocole TinyOS Beaconing. Périodiquement, la SB construit un arbre couvrant en diffusant un message RREQ ayant le format suivant :

$$i \rightarrow * : i, E(BK_i, ows \mid\mid i) \tag{3.4}$$

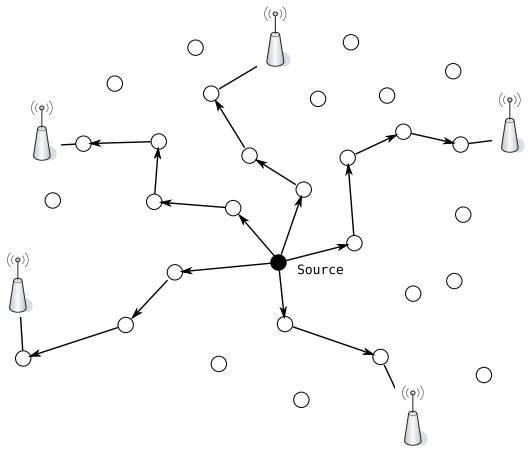


FIG. 3.4: Emulation d'une topologie multi-chemins à l'aide de plusieurs stations de base

Dans ce message, le numéro de séquence à sens unique (ows), ainsi que l'identificateur de l'émetteur, sont chiffrés à l'aide de la clé de broadcast BK_i. Cette clé est partagée entre un nœud et ses voisins atteignables grâce à un échange défi-réponse garantissant l'existence d'un lien bidirectionnel. Ce chiffrement du numéro de séquence ows vise donc deux objectifs : l'authentification de la SB et celle aussi du voisin atteignable. Quand un nœud j reçoit ce message, il décrypte la partie secrète et vérifie s'il s'agit d'un nouveau tour valide. Dans ce cas, le nœud j choisit i comme son parent, et ignore toutes les autres RREQ du même tour. Après un certain moment d'attente, j relaye sa décision en reconstruisant le message (3.4) et en le chiffrant avec sa clé BK_j.

3.2.1 Analyse

L'inconvénient principal d'EINSENS est sa faible tolérance aux pannes en présence d'une seule SB. Avec un chemin seulement vers la SB, la probabilité de déconnexion de cette route reste importante, et une reconstruction plus fréquente est nécessaire pour rafraîchir la vue de routage des nœuds. Ainsi, le gain en messages, obtenu grâce à la simple construction d'un seul chemin, peut être perdu avec cette reconstruction plus fréquente.

De plus, EINSENS n'est pas victime seulement d'un problème de tolérance aux pannes. Ce protocole est vulnérable à une attaque de spoofing durant la phase de construction des routes. Un intrus peut facilement relayer une RREQ correcte en utilisant l'identificateur d'un autre nœud voisin. Cela est possible à cause d'une erreur de conception des auteurs qui ont choisi un mauvais type de clé pour authentifier ces messages à un saut. En effet, EINSENS utilise des clés de broadcast symétriques partagées entre un nœud et ses voisins directs. Comme les deux extrémités de cette connexion doivent avoir la même clé, chaque nœud connaît les clés de broadcast de tous ses voisins à un saut. Par conséquent, un intrus peut utiliser les clés de ses voisins afin d'endosser leur identité et l'utiliser pour fabriquer des RREQ en leur nom.

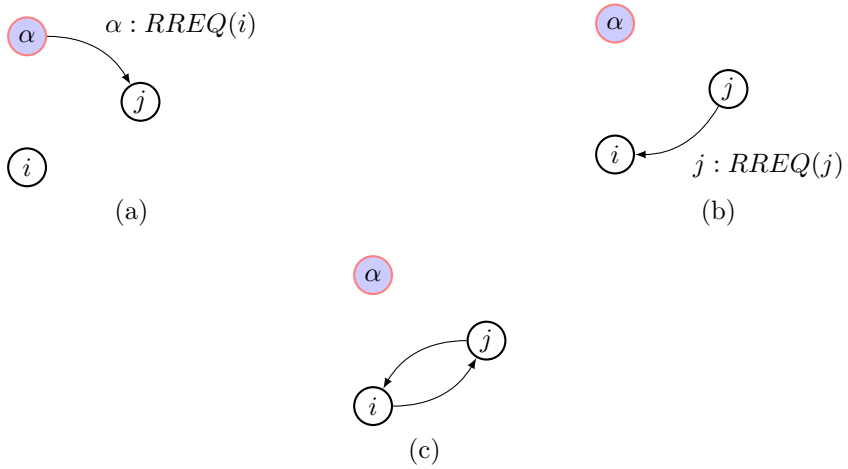


Fig. 3.5: L'attaque de spoofing utilisée contre EINSENS afin de créer une boucle de routage. (a) L'intrus α injecte la RREQ avec l'identité de i. Ainsi, j pense que i est son parent. (b) Le nœud j relaie le message, et donc i le choisit comme parent. (c) Les nœuds i et j se sont choisis mutuellement comme parent, créant ainsi une *boucle de routage*.

Donnons un exemple d'application de cette faille en essayant de créer une boucle de routage entre deux voisins. La Fig. 3.5 montre un scénario de cette attaque, où le symbole $x : RREQ(y)$ signifie que le nœud x envoie la RREQ en utilisant l'identité de y (ce qui signifie que x connaît la clé BK_y).

Lorsque l'attaquant α reçoit une RREQ valide de l'un de ses parents, il l'envoie en la chiffrant avec la clé BK_i de i. Le nœud j reçoit ce message, l'authentifie et rajoute i comme parent potentiel. Par la suite, j relaie le message RREQ avec sa propre identité, qui va être reçu par i. Ce dernier choisira donc j comme parent potentiel. Donc, i et j se choisiront mutuellement comme parent, créant ainsi une boucle de routage. Une boucle de routage, malgré sa simplicité, est fatale pour un protocole de routage, car elle fonctionne comme un trou noir aspirant tous les messages passant par des nœuds ligitimes, puisque aucun message ne pourra sortir de la boucle jusqu'à ce que son TTL soit nul. Notons que cet exemple n'est qu'un scénario possible, puisque le choix des parents est pseudo-alétoire, ce qui rend cette technique une attaque probabiliste. Toutefois, l'attaquant peut augmenter la probabilité de réussite en jammant les autres RREQ afin d'en laisser que les messages échangés entre les nœuds victimes.

3.3 SeRINS

SeRINS (*Secure alternate path Routing IN Sensor networks*) [22] fait partie des protocoles tolérants aux intrus et aux pannes grâce à sa topologie multi-chemins sécurisée, tout comme INSENS. Sa contribution majeure est sa protection semi-distribuée de la métrique du nombre de sauts à l'aide d'un ensemble de chaînes de hachage. Trois catégories de chaînes

sont employées dans SeRINS :

- Comme pour les protocoles précédents, une chaîne est utilisée pour l'authentification de la SB et les initialisations des nouveaux tours.
- Une chaîne générée « *à la volée* » durant le relais de la RREQ afin d'empêcher un intrus de décrémenter la valeur de son nombre de sauts réel. Cette chaîne, appelée nI, est la *preuve de distance* utilisée afin de mapper le nombre de sauts à une succession d'appels d'une fonction F sur une graine aléatoire (voir section 2.4.1.3).
- La chaîne nI ne peut détecter que les décrémentations du nombre de sauts. Un attaquant peut donc relayer la preuve de distance nI de son parent, faisant croire qu'il est à la même distance. Pour cela, SeRINS a rajouté une seconde chaîne, appelée nII, permettant de « *greffer* » l'identité du nœud relais avec sa distance afin d'éviter que ce nœud utilise la preuve de distance d'un autre nœud.

3.3.1 Sélection du parent principal

Initialement, la SB émet une RREQ avec un nombre de sauts nul et un nI aléatoire :

$$SB \rightarrow * : SB, h = 0, nI = r, nII = 0, ows \tag{3.5}$$

Quand un capteur détecte un nouveau tour (grâce au numéro de séquence à sens unique ows), il entre dans une phase préliminaire afin de découvrir son *parent principal* avant de relayer le message RREQ. Ce parent est le voisin ayant le plus petit nombre de sauts parmi les voisins atteignables. Après cette phase de sélection, le nœud relaie le message RREQ comme suit :

$$i \rightarrow * : i, h = h_p + 1, nI = F(nI_p), nII = F(i \mid\mid nI_p), ows \tag{3.6}$$

où h_p, nI_p représentent les valeurs reçues du parent choisi.

3.3.2 Sélection des parents alternatifs

Après ce relais, i débute une phase de découverte des chemins alternatifs. Quand le nœud i reçoit les RREQ ultérieures de la forme $\langle j, h_j, nI_j, nII_j, ows \rangle$ indiquant le même tour, il doit effectuer deux tests pour accepter j comme parent alternatif :

- Si $h_j \leq h_p$, le nœud i vérifie que $nI_p = F^{h_p - h_j}(nI_j)$. Ce test sera satisfait si nI_j et nI_p ont été générés de la même graine, puisque $F^{h_p}(r) = nI_p$ et $F^{h_j}(r) = nI_j$.

– Si $h_j = h_p + 1$, les nœuds i et j sont à la même distance de la SB. Par conséquent, ils doivent partager la même valeur de h_p. Pour vérifier que j n'a pas menti en relayant la valeur de nI de son parent sans l'incrémenter avec F, le nœud i utilise le champ nII. Comme nI du parent de j doit être égal à nI_p de i, i peut recalculer la vraie valeur de nII_j puisque $nII_j = F(j \mid\mid nI_p)$.

Si l'un de ces tests échoue, le nœud i émet un message d'alerte à la SB. Afin de garantir que l'attaquant ne bloquera pas l'acheminement de ce message, SeRINS l'envoie en utilisant une diffusion générale dans tout le réseau. Après que la SB authentifie cette alerte, elle interroge les voisins du nœud suspect afin de collecter plus d'informations. Si les valeurs de nombre de sauts et les différentes valeurs de chaînes sont consistantes, le nœud suspect est déclaré comme compromis. Dans le cas contraire, la SB déduit que l'alerte initiale est fausse et déclare le nœud émetteur de l'alerte comme compromis.

3.3.3 Analyse

Ce protocole se focalise principalement sur la protection de la métrique du nombre de sauts. Malgré que l'idée de base repose sur une bonne conception, mais elle recèle une faille importante. Pour bien comprendre cette faille, analysons l'origine de l'idée.

L'idée d'utiliser une chaîne de hachage pour protéger la métrique du nombre de sauts a déjà été employée par quelques protocoles MANET [12]. La différence principale est que ces derniers utilisent une cryptographie à clé publique afin de protéger la graine de génération initiale, car cette dernière est la valeur centrale permettant de vérifier le mapping entre le nombre de sauts déclaré et le nombre de hachages effectués. En utilisant une signature sur cette valeur, chaque nœud du réseau ad hoc peut vérifier l'intégrité de cette valeur, ce qui lui permet facilement de détecter toute tentative de décrémentation du nombre de saut par l'un de ses voisins.

Pour adapter cette solution aux environnements WSN, les auteurs de SeRINS ont été obligés de se débarrasser de la cryptographie à clé publique, perdant ainsi la protection de la graine de génération de la chaîne. Pour remédier à ce manque de protection, ils ont choisi de comparer les chaînes reçues entre elles afin de détecter les éventuelles incohérences, car seule une graine valide leur permet d'effectuer la vérification d'une valeur de hachage d'une manière indépendante. Cette comparaison se base sur un référentiel important, qui est le parent principal. Ce dernier permet détecter les hachages qui n'ont pas été générés avec la même graine utilisé par le chemin du parent principal. Or, la sélection de ce dernier se fait d'une manière totalement non-sécurisée. Ainsi, le référentiel permettant de détecter les éventuelles incohérences est lui même sujet à l'incohérence !

Par conséquent, un intrus peut injecter, pendant la phase de sélection du parent principal, des RREQ avec un hop count falsifié, ce qui va être *aveuglement* accepté par ses voisins. Comme cette valeur erronée sera utilisée pour vérifier les futures RREQ, toutes ces vérifications seront erronées, produisant de fausses alertes qui inculperont des nœuds légitimes.

3.4 SecRout

Le protocole SecRout [46] est un protocole de routage hiérarchique sécurisé. Le réseau est organisé en clusters ayant chacun un chef. Le nœud puits est supposé connaître cette organisation du réseau, et doit maintenir localement une table contenant une clé secrète de chaque capteur. Cette clé est supposée pré-chargée dans chaque nœud. De plus, chaque cluster doit posséder une clé permettant de sécuriser les échanges intra-cluster. Cette clé doit être connue par le clusterhead et tous les nœuds du groupe. Le protocole SecRout ne spécifie pas l'algorithme de construction de clusters, et suppose que les clusters ainsi que leurs clés sont établis par un autre protocole, comme LEAP [48].

Contrairement aux protocoles précédents, SecRout est basé sur une topologie mesh établie grâce à un schéma RREP-RREQ initié par les nœuds capteurs. SecRout est donc composé de deux phases : propagation de la requête et relais de la réponse.

3.4.1 Propagation de la requête

Le nœud source S initie une découverte des chemins en émettant vers ses voisins directs un paquet de requête RREQ contenant les informations suivantes :

$$\begin{array}{c} S \mid\mid ID_{SB} \mid\mid ID_{RREQ} \mid\mid N_{source} \\ MAC\left(K_S, S \mid\mid ID_{SB} \mid\mid ID_{RREQ} \mid\mid N_S\right) \end{array} \tag{3.7}$$

avec ID_{RREQ} est l'identificateur de la requête.

Lorsqu'un nœud reçoit une requête, elle n'est acceptée qu'avec l'unicité de son identificateur ID_{RREQ}. Il met à jour par la suite sa table de routage en utilisant l'information des deux sauts précédents vers la source. Avant de relayer la requête, le nœud remplace les valeurs de ID_{pre} et ID_{this} par, respectivement, ID_{this} et son identificateur :

$$\begin{array}{c} ID_{this} \mid\mid ID_{pre} \mid\mid S \mid\mid ID_{SB} \mid\mid ID_{RREQ} \mid\mid N_S \\ MAC\left(K_S, S \mid\mid ID_{SB} \mid\mid ID_{RREQ} \mid\mid N_S\right) \end{array} \tag{3.8}$$

3.4.2 Relais de la réponse

Lorsque la SB reçoit la première requête, il vérifie le MAC construit par la source en utilisant la clé relative à son identificateur S. Si le MAC est correct, la SB met à jour sa table de routage en utilisant les champs ID_{pre} et ID_{this}.

La SB génère ensuite une réponse RREP ayant le format suivant :

$$\begin{gathered} ID_{pre} \mid\mid ID_{this} \mid\mid ID_{next} \mid\mid S \mid\mid ID_{RREQ} \mid\mid N_S \\ MAC\left(K_S, S \mid\mid ID_{SB} \mid\mid ID_{RREQ} \mid\mid N_{SB}\right) \end{gathered} \tag{3.9}$$

La requête est émise en broadast locale, ciblé à l'aide du champs ID_{next}. Lorsque le capteur voisin ayant l'identificateur ID_{next} reçoit cette réponse, il met à jour sa table de routage en conséquence[1], puis remplace les champs ID_{pre} et ID_{this} par ID_{this} et son identificateur. Il doit aussi modifier le champ ID_{next} par l'identificateur connus lors de la phase de découverte de chemins.

Si le nœud ne reçoit pas la réponse émise par ID_{next} après un certain temps, il ignore toutes les requêtes émises pendant la prochaine phase de découverte.

Le nœud de relais doit aussi vérifier que la réponse émise par le prochain saut est valide, en s'assurant qu'elle est bien destinée au nœud à deux sauts contenu dans le chemin vers la source.

Lorsque la source reçoit la première réponse, elle vérifie le MAC généré par le puits et met à jour sa table de routage en ajoutant ID_{this} et $IDpre$ comme prochains sauts vers le puits.

3.4.3 Analyse

En essayant d'alléger les calculs cryptographiques pour préserver l'énergie des nœuds, SecRout a omis un point important pour sécuriser la construction des routes : l'authentification du prochain saut. Ce protocole n'inclut effectivement qu'une authentification point à point entre la source et la destination. Les nœuds intermédiaires ne procèdent à aucun contrôle cryptographique sur les messages RREQ/RREP échangés. Cette lacune ouvre la possibilité à plusieurs vulnérabilités, dont les plus importantes seront décrites par la suite.

Une attaque possible est d'empoisonner le cache de routage des nœuds relais avec de fausses informations. Cela est rendu possible parce que les nœuds intermédiaires, relayant un message RREQ, doivent sauvegarder les informations d'identification du chemin afin de l'utiliser lors du relais du message RREP. Un attaquant peut donc injecter un nombre important de messages RREQ avec des identités aléatoires, empoisonnant le cache des voisins par des routes inexistantes.

Une autre attaque possible est le Hello Flooding. Lorsque l'intrus relais un message RREQ correcte, il utilise une grande puissance d'émission avec l'identificateur d'un nœud légitime. Puisqu'aucune authentification n'est employée à ce niveau, cela résultera en un ensemble de liens totalement incohérents. En plus, comme SecRout utilise un mécanisme de watch-dog lors afin de vérifier le comportement de l'émetteur du message RREQ lors du relais du RREP, le nœud légitime, dont l'identité a été utilisé par l'intrus, se retrouve suspect car il ne peut pas effectuer le relais.

[1] En utilisant ID_{pre} et ID_{this} afin de remplir le champ $Next$ de la table de routage.

3.5 Conclusion

Ce chapitre a présenté les détails de fonctionnement et la philosophie de conception des solutions de dissémination fiable les plus représentatives dans les WSN. À cause des besoins et contraintes de ces réseaux, les topologies de collection, basées sur des arbres, sont les plus adéquates pour ce paradigme de communication. Cela est principalement expliquée par la position centrale qu'occupe la station de base, qui est la destination commune de tous les capteurs. Les solutions sécurisées présentées sont aussi caractérisées par l'emploi exclusif du chiffrement symétrique, qui reste actuellement la solution la plus économe en énergie afin de protéger les données contre les éventuelles attaques.

Toutefois, l'analyse des protocoles nous permet de conclure que ces solutions souffrent d'un déséquilibre entre le niveau de fiabilité offert et la scalabilité du système. En effet, d'une part, les solutions offrant une solide sécurité emploient un échange excessif de messages afin d'établir cette structure fiable. Et d'autre part, les solutions *légères* en consommation n'offrent qu'une tolérance limitée aux pannes. Dans le prochain chapitre, cette problématique ainsi que sa solution seront étudiées à travers notre proposition : le protocole SEIF.

Chapitre 4

Notre Solution : SEIF

4.1 Introduction

Les problèmes des pannes naturelles et des attaques d'intrus sont deux concepts partageant en commun plusieurs ressemblances. En effet, ces deux problématiques peuvent être regroupées comme étant des problèmes de défaillance pouvant générer certains comportements byzantins. Le but est de relayer les données via des chemins non infectés par des nœuds présentant de tel comportements. C'est pour cette raison que plusieurs solutions présentées précédemment traitent ces deux problèmes d'une manière *unifiée* en construisant une topologie mutli-chemins sécurisée. Cette approche couplée permet d'offrir deux niveaux de protection :

- *Une défense réactive* lors de la construction des routes, et cela contre les tentatives d'attaques. Cette défense se base sur l'utilisation de primitives cryptographiques pour implémenter des objectifs de sécurité, suivant la nature de l'information à protéger.
- *Une défense préventive*, utile lors du relais des données, se base sur la topologie multi-chemins afin d'offrir une *duplication du relais* et éviter les nœuds défaillants.

Dans la littérature existante, l'idée de fusionner les problèmes de sécurité et de tolérance aux pannes est principalement illustrée par deux protocoles : INSENS et SeRINS. Ces deux protocoles ont adopté des approches assez différentes qui souffrent des inconvénients suivants :

- INSENS souffre principalement de sa lourdeur, causée par son approche centralisée. À chaque construction des routes, un nombre important de messages de contrôle sont échangés entre les motes et la SB, ce qui fait qu'INSENS ne peut être appliqué à des réseaux de grande dimension. Néanmoins, l'avantage primaire d'INSENS est qu'il offre un contrôle total sur la qualité de la topologie de communication et sa fiabilité.
- SeRINS essaye d'outrepasser l'inconvénient d'INSENS en implémentant une détection

semi-distribuée qui ne fait appel à la SB qu'en cas de comportements suspects. Toutefois, ce protocole a choisi de sacrifier la qualité de la topologie multi-chemins en omettant la propriété de routes disjointes. Ce choix fait que la topologie devient moins tolérante aux pannes et aux intrus vu la diminution du taux de duplication.

Par conséquent, on peut dire que les solutions existantes n'offrent pas de *compromis* intéressant entre le niveau de *tolérance* du système et sa *scalabilité*. Les topologies à chemins disjoints, qui sont les plus résistantes aux défaillances, sont construites avec un nombre excessif de messages. Alors que les solutions scalables offrent une topologie multi-chemins assez fragile en présence de pannes.

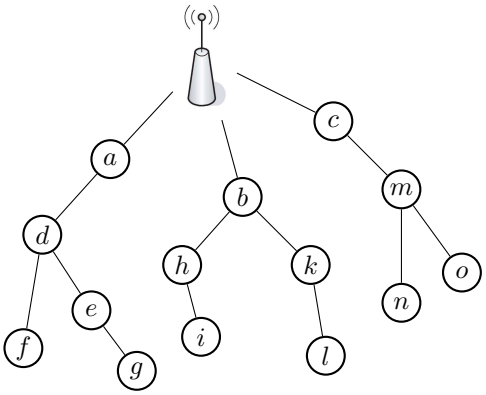
Dans ce chapitre, nous allons décrire notre nouvelle solution, appelée SEIF (*Secure and Efficient Intrusion-Fault tolerant routing protocol for wireless sensor networks*) [27], palliant à ces différents problèmes en présentant une architecture *totalement distribuée* capable de construire des chemins disjoints d'une manière très *scalable*. En plus, nous présenterons aussi une version étendue de SEIF permettant d'offrir une meilleure tolérance via une amélioration de la notion de routes disjointes. En relâchant certaines contraintes, la version étendue de SEIF permet d'augmenter la tolérance de la topologie tout en préservant la scalabilité du système.

4.2 Idée de base

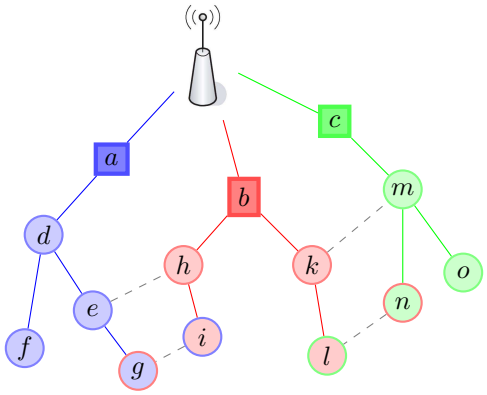
Dans la section 2.4.3, on a expliqué que la tolérance d'une topologie est principalement définie par le taux de duplication offert par les chemins alternatifs. Cette duplication est étroitement liée au taux de chevauchement des routes. C'est pour cette raison que les chemins disjoints représentent une solution très attrayante à cause de l'intersection nulle entre les chemins. L'absence d'intersection offre donc une duplication totale de la donnée durant le relais.

Il existe dans la littérature plusieurs méthodes d'établissements de chemins disjoints [25, 5, 45]. Parmi ces solutions, *la méthode des branches* apparaît comme étant l'alternative la plus intéressante, puisqu'elle n'emploie qu'un seul message par nœud pour découvrir ces routes disjointes. De ce fait, elle ne rajoute aucun overhead sur la découverte simple d'arbres classiques, comme par exemple le protocole TinyOS Beaconing.

Cependant, les solutions existantes ne proposent aucun mécanisme de sécurité afin de protéger la construction d'une telle topologie. Avant de présenter notre solution à ce problème, nous allons décrire brièvement le mécanisme des branches et ses différentes failles.



(a) Arbre classique uni-chemin



(b) Arbre multi-chemins basés sur la notion de racines (nœuds carrés) et de branches (différenciées par les couleurs)

Fig. 4.1: Le concept de branches. (a) Une topologie en arbre classique produite par des protocoles tels que TinyOS Beaconing. (b) En utilisant le concept de branches, la réception redondante des messages de construction peut être exploitée afin de découvrir de nouveaux chemins, sans rajouter de messages additionnels. Par exemple, quand le nœud g émet son message RREQ, le nœud i peut découvrir un chemin alternatif à travers la branche bleue, puisque i appartient déjà à la branche rouge.

4.2.1 Les branches

Le principe de ce mécanisme est assez simple et se base sur la notion de *racines* et de *branches* [25]. Une racine est un voisin direct de la SB, et son sous-arbre est appelé branche. Les messages RREQ sont « *marqués* » avec l'identificateur de la racine ayant relayé le message (voir Fig. 4.1). Cette information étant non mutable, elle doit rester inchangée durant tout le relais du message. Lorsqu'un nœud reçoit plusieurs RREQ, il doit *filtrer* ces messages afin d'en laisser que ceux ayant parcouru des chemins disjoints. Pour ce faire, le nœud utilise la marque de chaque message pour distinguer les différentes branches : *deux routes appartenant à des branches différentes sont forcément disjointes*. Cela découle du fait que chaque mote ne peut émettre qu'une seule RREQ qui ne contient qu'une seule marque. Cette marque est l'identificateur de sa branche principale choisie (aléatoirement) parmi les branches découvertes. Ainsi, deux RREQ marquées avec des branches différentes ne peuvent pas être émises par le même nœud. Par conséquent, les routes ayant acheminé ces deux RREQ ne partagent aucun nœud en commun.

D'après cette courte description, on remarque qu'une topologie multi-chemins disjoints peut être établie avec seulement un seul message par nœud et avec une faible complexité de stockage, ce qui fait le point fort de cette approche.

4.2.2 Vulnérabilités

Malheureusement, malgré ses avantages nombreux, ce mécanisme de branches comporte des failles de sécurité importantes. Ces failles peuvent être classées suivant le type d'information ciblée :

- La *non-protection de l'identité de la racine* est une brèche très importante. En laissant cette information cruciale sans contrôle, un intrus peut intervenir dans la décision des nœuds de son voisinage afin de les obliger à le choisir comme prochain saut. En effet, le fait de baser le choix des routes sur l'identificateur des racines rajoute du *déterminisme* au mécanisme de décision des nœuds, ce qui offre une opportunité intéressante pour l'intrus. Par exemple, en injectant des branches inexistantes, les voisins de l'intrus vont automatiquement le choisir comme parent (principal ou alternatif), car une branche inexistante est forcément *disjointe* de toute autre branche existante.

- Comme la construction de la topologie peut être réitérée plusieurs fois, il est nécessaire d'identifier chaque RREQ avec le numéro du tour courant, ce qui fonctionnera comme estampille du message et permettra de distinguer les routes fraîches des anciennes déclarations. Une propriété fondamentale de cette estampille est qu'elle permet de réinitialiser (*i.e.* effacer) la table de routage des nœuds. En effet, lorsqu'un nœud reçoit une RREQ appartenant à un nouveau tour, il considère que sa vue sur la topologie de routage est devenue « ancienne », et doit donc la supprimer afin de la mettre à jour.
  Par conséquent, l'injection d'une RREQ avec un nouveau tour placera l'intrus comme étant le seul prochain saut de ses voisins, car sa RREQ sera considérée comme l'information la plus fraîche, et donc la plus prioritaire. En plus, cette information va se propager dans tout le réseau et toutes les routes seront dirigées vers l'intrus, créant ainsi un *sinkhole*.

- La dernière information à protéger est l'identité du prochain saut. Sans protection à ce niveau, l'intrus peut :
  - Relayer une RREQ en utilisant de fausses identités.
  - Utiliser l'identificateur de nœuds voisins afin de créer des boucles de routage.
  - Attaquer en utilisant sa propre identité, mais en augmentant sa portée de communication. Ainsi, l'attaquant peut créer avec une RREQ valide des liens unidirectionnels, ce qui est connu sous le nom d'attaques Hello Flooding. Certains nœuds touchés par ce message choisiront l'intrus comme prochain saut, sans pour autant pouvoir communiquer avec lui vu le lien unidirectionnel.

4.2.3 Notre solution

Dans la littérature existante, aucune solution n'existe pour sécuriser ce mécanisme. Dans ce qui suit, nous allons présenter notre solution SEIF permettant d'établir efficacement et d'une manière sécurisée une topologie multi-chemins à chemins disjoints basée sur le concept

de branches.

Le problème principal du mécanisme basique de branches est l'utilisation des identificateurs des racines en tant que marque de branches. Cette information « *en clair* » permet à un attaquant de manipuler librement la vue de ses voisins et leur décisions. Vue la présence éventuelle d'intrus internes, il est impossible d'utiliser un simple chiffrement pour protéger cette information, car les clés secrètes seront connues par l'intrus interne. Dans ce cas, le seul mécanisme efficace pouvant protéger les marques de branches est l'*authentification d'origine*. En effet, si on peut assurer que l'identificateur de la branche présumée provient réellement d'un *voisin direct* de la SB, un attaquant ne pourra plus injecter de fausses branches. Cependant, cette authentification doit vérifier les propriétés suivantes :

- Cette authentification doit suivre le modèle *un-à-plusieurs*, où la racine doit pouvoir produire une preuve vérifiable par n'importe quel capteur du réseau. Autre que la racine concernée, aucun autre capteur ne doit pouvoir générer à l'avance cette preuve, ce qui nécessite une certaine propriété *asymétrique* rendant le mécanisme à sens unique.
- Elle doit être dynamique afin de résister aux attaques de re-jeu. Cela est nécessaire lorsque la topologie de routage doit être rafraîchie périodiquement. Ainsi, les preuves d'authentification générées lors d'un tour i, doivent être invalides pour tout tour $j > i$.
- La solution choisie doit prendre en considération les contraintes de motes en conservant l'énergie et l'espace mémoire.

Afin d'assurer ces trois propriétés, notre solution repose sur le concept de chaînes de hachage [1]. Ce mécanisme permet d'offrir les propriétés requises, qui sont : *l'asymétrie*, *le dynamisme* et *la préservation des ressources*.

Comme décrit précédemment, une authentification basée sur une chaîne de hachage nécessite que l'entité authentifiante possède au préalable une graine d'authentification de la chaîne. Dans le cas des branches, chaque capteur du réseau doit connaître une telle graine pour chaque racine. Mais l'utilisation usuelle des chaînes est plus simple, car le récepteur connaît au préalable l'entité à authentifier. Dans notre cas, les entités à authentifier, *i.e.* les racines, ne sont connues qu'*après* le déploiement du réseau, et leur nombre peut *changer* au cours du temps avec l'ajout/départ des nœuds. Par conséquent, on ne peut pas établir une association de sécurité *statique* entre les racines et le reste du réseau.

Pour contourner ce problème, nous avons utilisé la SB comme repère commun entre les racines et le reste du réseau. Avant le déploiement, un certain nombre de chaînes sont générées et sauvegardées au niveau de la SB. Chaque capteur est pré-chargé avec la première valeur non utilisée de chaque chaîne. Au début de chaque tour, la SB distribue pour chaque racine i sa « *marque de branche* » en divulguant la prochaine valeur d'une chaîne donnée. Cette marque sera utilisée par la racine afin de construire sa branche et prouver au reste du réseau son voisinage directe avec la SB, puisque *seule* la SB peut connaître la prochaine valeur d'une chaîne. Lorsqu'un capteur reçoit une telle marque, il pourra vérifier son appartenance à une

[1] On peut utiliser la cryptographie à clé publique, mais elle n'est pas adéquate aux WSN

des chaînes de la SB, puisqu'il possède une valeur antécédente de la chaîne en question.

Notons que ce mécanisme découple l'identification de la branche et l'identification de la racine. En effet, la SB distribue à chaque tour un nombre de « *tickets* » utilisés pour marquer les branches valides. Ainsi, la SB contrôle la borne maximale du nombre de branches valides durant un tour, empêchant l'intrus d'injecter de fausses branches.

La seconde information à protéger est l'identification des tours. Il faut s'assurer que seule la SB possède le privilège d'initier un nouveau tour. Pour cela, on a besoin aussi d'un mécanisme de vérification asymétrique des incrémentations des tours. SEIF utilise une chaîne de hachage comme générateur de numéros de séquence à sens unique. À chaque nouveau tour, la SB révèle, dans le sens inverse de génération, la prochaine valeur de la chaîne dans la RREQ initiale. Comme les capteurs possèdent la première valeur de la chaîne, ils peuvent vérifier l'appartenance du nouveau numéro de séquence à la chaîne des tours.

La dernière information à sécuriser est le prochain saut de la route. Dans ce cas, il faut authentifier l'émetteur du dernier relais de la RREQ et vérifier son appartenance au voisinage *atteignable*. SEIF fusionne deux concepts pour atteindre ces deux objectifs :

- Chaque capteur génère une chaîne de hachage *locale*, qui sera utilisée pour authentifier les broadcasts locaux à un saut.
- La graine d'authentification de cette chaîne est divulguée *seulement* aux voisins atteignables. Pour ce faire, SEIF utilise une clé de broadcast établie avec des mécanismes tels que LEAP. Ce genre de protocoles utilise des techniques défi-réponse, garantissant l'existence d'un lien bi-directionnel entre les voisins en question. Par conséquent, SEIF chiffre la graine d'authentification avec cette clé de broadcast, qui ne sera divulguée qu'aux voisins atteignables.

4.3 Description du protocole

4.3.1 Initialisation

SEIF comporte deux types d'initialisations : globale et locale. L'initialisation globale est effectuée avant le déploiement du réseau en générant au niveau de la SB deux types de chaînes :

- Un ensemble de chaînes $(BC_i)_{0 \leq i \leq n}$ – *Branch Chains* – employées pour l'authentification des racines. Le nombre de chaînes correspond au nombre maximal de branches durant un seul tour.
- Une chaîne R est nécessaire pour authentifier les initialisations des nouveaux tours.

Chaque capteur déployé i effectue son initialisation locale, qui se résume en trois actions :

- Pour chaque chaîne BC_j, le nœud i maintient un vérificateur $BV_{i,j}$ – *Branch Verifier* – initialisé avec la première valeur non utilisée de la chaîne. En plus, i doit aussi maintenir

une variable $P_{i,j}$ reflétant la position de $BV_{i,j}$ au sein de la chaîne BC_j.[2]

- Un autre vérificateur RV_i – *Round Verifier* – est créé pour la chaîne R afin d'authentifier les tours.
- Pour chaque voisin atteignable j, un vérificateur de voisinage $NV_{i,j}$ – *Neighborhood Verifier* – est maintenu. Pour cela, le capteur génère localement sa propre chaîne d'authentification de voisinage NC_i . Après l'établissement de la clé de broadcast BK_i, le nœud i divulgue à ses voisins la première valeur non utilisée V de la chaîne NC_i :

$$i \rightarrow * : i, E(BK_i, V) \tag{4.1}$$

Lorsqu'un nœud j reçoit ce message, il initialise son vérificateur de voisinage $NV_{j,i}$. Si le nœud j n'est pas un nœud nouvellement déployé (*i.e.* il a achevé sa phase d'initialisation) et i représente un nouveau voisin de j, ce dernier devra répondre en envoyant la dernière valeur de sa chaîne NC_j chiffrée avec sa clé de broadcast BK_j.

4.3.2 Distribution des marques

Le but de cette étape est de fournir à chaque racine sa marque d'authentification. Au début de chaque tour, la SB prélève le prochain numéro de séquence r du tour actuel à partir de la chaîne R, et assigne à chaque racine i la prochaine valeur d'une des chaînes BC. Notons que cet assignement est aléatoire, et peut varier d'un tour à un autre. Ces informations sont véhiculées avec le message suivant :

$$SB \rightarrow i : E(K_i, n \mid\mid V \mid\mid P \mid\mid r) \tag{4.2}$$

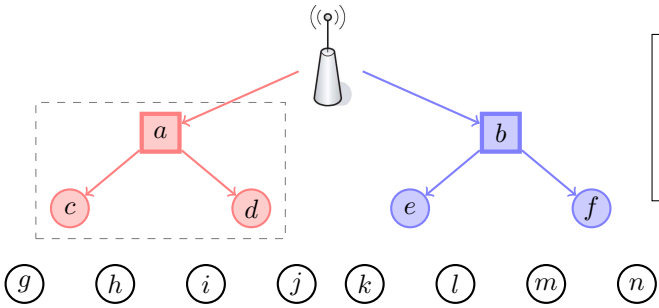
où :

- K_i est la clé secrète de la racine i, connue seulement par la SB et i.
- n est le numéro de la chaîne affectée à la racine i pendant ce tour.
- V est la première valeur non utilisée de la chaîne BC_n.
- P est la positon de V au sein de la chaîne BC_n.
- r est le numéro de séquence du tour courant prélevé de la chaîne R.

Après déchiffrement du message, le nœud i vérifie la validité du nouveau numéro de séquence en vérifiant que $RV_i = F(r)$. Si cette relation est correcte, le vérificateur RV_i est mis à jour avec la nouvelle valeur reçue. Par la suite, le nœud i vérifie la validité de la nouvelle marque à l'aide de ces deux relations :

$$\begin{cases} P > P_{i,n} \\ BV_{i,n} = F^{P-P_{i,n}}(V) \end{cases}$$

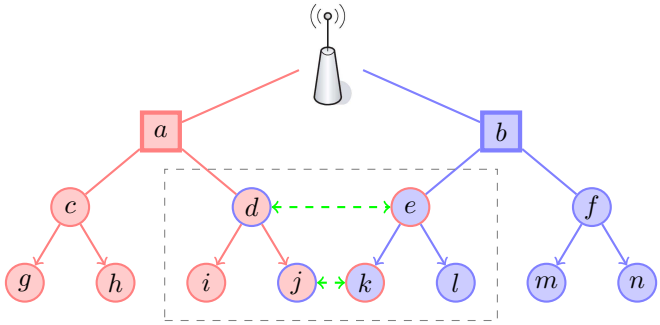
[2]Toutes ses informations sont pré-chargées lors de la programmation du mote avec le code de l'application, et cela avant son déploiement.



$$sink \rightarrow a : E(K_a, n_1||V_1||P_1||r)$$

$$a \rightarrow * : a, n_1, V_1, P_1, R, NP_a$$

(a) Cette figure décrit la distribution des marques au sein de la branche délimitée par le rectangle en pointillées. Le même processus est exécutée dans la seconde branche du nœud b. La SB commence par envoyer à la racine a sa marque V_1, prise d'une des chaînes (BC_i). Le nœud a authentifie cette valeur, ainsi que le numéro de séquence r. Par la suite, la racine a commence la construction de sa branche en envoyant ces informations authentifiées aux voisins c et d.



$$d \rightarrow * : d, n_1, V_1, P_1, r, NP_d$$
$$e \rightarrow * : e, n_2, V_2, P_2, r, NP_d$$
$$j \rightarrow * : j, n_1, V_1, P_1, r, NP_j$$
$$k \rightarrow * : k, n_2, V_2, P_2, r, NP_k$$

(b) La construction commence donc par l'émission des messages RREQ par les racines a et b. Lorsqu'un capteur reçoit un tel message, il effectue trois types de vérifications : l'authentification du parent, l'authentification de la branche et la vérification du tour. Par exemple, quand le nœud i reçoit le message de d (appartenant à la branche de a), i vérifie les équations suivantes : (1) $NV_{i,d} = F(NP_d)$, (2) $RV_i = F(R)$ or $R = RV_i$, (3) $BV_{i,n_1} = F^{P_1 - P_{i,n_1}}(V_1)$. Les liens en pointillés illustrent les chemins alternatifs disjoints découverts entre des nœuds appartenant à des branches distinctes.

Fig. 4.2: Un exemple d'exécution de SEIF

Ces deux relations garantissent l'authenticité de la valeur V en vérifiant son appartenance à la chaîne BC_n. Les variables $P_{i,n}$ et $BV_{i,n}$ sont mises à jour avec les nouvelles valeurs reçues.

4.3.3 Construction de la branche

Après réception des marques valides, chaque racine i commence la construction de sa branche en émettant le message suivant :

$$i \rightarrow * : i, n, V, P, R, NP \tag{4.3}$$

où :

- n, V, P et r sont les informations reçues dans le message (4.2).
- NP est la première valeur inutilisée de la chaîne locale NC_i.

Lorsqu'un capteur j reçoit le message (4.3), il authentifie l'émetteur en vérifiant si NP représente le prochain numéro de séquence de la chaîne NC_i, *i.e.* $NV_{j,i} = F(NP)$. Après une authentification réussie et mise à jour du vérificateur $NV_{j,i}$, le nœud j vérifie la marque de la branche en utilisant ces trois relations :

$$\begin{cases} P > P_{j,n} \\ BV_{j,n} = F^{P-P_{j,n}}(V) \\ P - P_{j,n} < D \end{cases}$$

Les deux premières relations sont identiques à celles utilisées par la racine. Nous avons rajouté la troisième condition afin d'éviter des attaques DoS visant à épuiser l'énergie des nœuds. En effet, sans cette condition, un intrus peut relayer une RREQ valide mais avec une grande valeur de P afin de forcer ses voisins à exécuter inutilement la deuxième relation avec un grand nombre d'itérations. Ainsi, la constante D permet d'établir une borne maximale sur le nombre d'itérations possibles à exécuter.

Après vérification de la marque et mise à jour de $P_{j,n}$ et $BV_{j,n}$, le nœud j vérifie le numéro de séquence du tour. Contrairement aux racines, le nœud j doit vérifier deux cas :

- Si $BV_i = F(r)$, j entre dans un nouveau tour en supprimant toutes les anciennes routes déjà découvertes et insère le nœud i comme prochain saut vers la branche numéro n. Le nœud j doit aussi déclencher un timer aléatoire définissant sa période de découverte des routes.

- Si $BV_i = r$, le nœud j conclut que cette RREQ appartient au tour courant et doit vérifier si le chemin reçu est disjoint de toutes les autres routes déjà découvertes. Puisque $BV_{i,n} = F(V)$, la branche n n'a pas été découverte pendant ce tour, signifiant que la route découverte est disjointe des autres routes déjà connues pendant ce tour.

Après expiration du timer aléatoire, le nœud j doit choisir sa branche principale parmi les routes découvertes. Ce choix ce fait d'une manière totalement aléatoire. Comme décrit dans le paragraphe 2.4.1.3, un tel choix *aveugle* permet de renforcer la sécurité en éliminant toute influence de l'intrus sur les décisions de ces voisins. Notons que SEIF implémente ce principe d'une manière plus efficace que EINENS, car ce dernier effectue un choix aléatoire *biaisé* reposant sur une politique FIFO. Cette dernière peut permettre à un intrus rapide d'influencer le choix de ces voisins. Par conséquent, SEIF utilise un choix *plus aveugle* que EINSENS, ce qui est considéré, dans un contexte de sécurité, plus solide.

Après le choix de la branche principale, j relaie son choix en utilisant le message (4.3). Avec ce relais récursif, chaque nœud du réseau appartiendra à l'arbre multi-chemins en n'utilisant qu'un seul message par mote.

4.4 Extension : SMRP

Dans cette section, nous allons décrire une extension du protocole SEIF, appelée SMRP (*Sub-branch Multi-path Routing Protocol*), permettant d'améliorer sa tolérance en adoptant une nouvelle variante de chemins disjoints.

4.4.1 Problématique

L'inconvénient majeur des chemins totalement disjoints est qu'ils ne sont pas très *abondants*, signifiant qu'un nœud peut ne pas découvrir assez de routes alternatives. La « rareté » de ces chemins est due principalement au fait que ce paradigme subit une contrainte assez forte stipulant que tout couple de chemins ne doivent partager *aucun* nœud en commun. La non abondance des chemins disjoints est un problème important, car la tolérance de la topologie dépend non seulement du taux d'intersection des routes (qui est dans ce cas optimal), mais aussi du nombre de routes découvertes.

Cet inconvénient a été soulevé auparavant dans le travail de Lee et al. [25], qui ont proposé une solution pour améliorer la tolérance offerte par les chemins disjoints. Ils ont remarqué que la découverte des branches ne s'effectuent pas d'une manière équitable entre les nœuds, faisant que l'information de certaines branches se propage mal dans certaines zones du réseau. En effet, la connaissance des chemins disjoints est seulement partagée entre des voisins appartenant à deux branches distinctes. Donc, deux nœuds d'une même branche ne partagent pas cette information, ce qui bloque la propagation de l'existence d'une branche x au sein des nœuds d'une autre branche y, sauf à sa périphérie.

La solution proposée par Lee et al. essaye de propager cette information en rajoutant plus des messages de contrôle, brisant ainsi la propriété de n'utiliser qu'un seul message par nœud. Lorsqu'un nœud découvre un nouveau chemin, il informe ses voisins sur la nouvelle branche. À l'aide d'une propagation récursive, cette information est diffusée vers les nœuds se trouvant au milieu de la branche. Il est clair que cette approche possède un inconvénient majeur, qui est la surcharge du réseau avec un nombre important de message de contrôle, diminuant la scalabilité du protocole et augmentant la consommation en énergie.

4.4.2 Idée de base de notre solution

Dans notre solution, on a choisi de préserver la contrainte de n'utiliser qu'un seul message par nœud. Afin de découvrir plus de chemins alternatifs, on a relâché la contrainte de disjointure totale des chemins sans pour autant altérer leur tolérance.

Dans les solutions existantes, un capteur rejette automatiquement tout chemin appartenant à une branche déjà découverte. Ainsi, un capteur ne peut accepter qu'un seul chemin par branche, ce qui limite son degré. Afin de découvrir plus de chemins, on a enlevé cette

contrainte en permettant certains nœuds particuliers d'être partagés par des chemins d'une même branche. Plus particulièrement, on tolère l'intersection des chemins au niveau des nœuds racines seulement. Ce choix est guidé par deux raisons :

- Le nombre de racines représente la borne maximale de chemins alternatifs possibles dans l'approche traditionnelle. Pour dépasser cette borne, on permet à un nœud d'accepter plusieurs chemins d'une même branche s'ils se croisent au niveau de la racine. Ainsi, au lieu de marquer les RREQ avec les identificateurs des racines (ou une autre marque propre à la racine, comme les valeurs de chaînes présentes dans SEIF), SMRP assigne la responsabilité de marquage aux voisins des racines, *i.e.* aux voisins à deux sauts de la SB. Ces nœuds deviennent donc des sous-racines qui vont construire leur sous-branches, dans lesquelles la règle de disjointure va être appliquée. Par conséquent, un capteur va accepter deux chemins de la même branche s'ils appartiennent à deux sous-branches distinctes. De cette façon, SMRP ne rejette pas aveuglement les chemins d'une même branche, ce qui permet de découvrir plus de chemins alternatifs tout en contrôlant le niveau d'intersection.
- La deuxième raison guidant ce choix est liée à la raison d'être des chemins disjoints. L'absence d'intersection permet de restreindre l'impact d'une défaillance sur un seul chemin au maximum. Cependant, on peut considérer que les nœuds racines sont des motes « spéciaux » ayant des caractéristiques (d'énergie et de protection) différentes des autres nœuds du réseau [3]. Ainsi, en tolérant l'intersection des chemins au niveau de ces nœuds, on ne perd pas en tolérance par rapport à la version des chemins disjoints, car la probabilité de déconnexion au niveau des nœuds racines est devenue assez faible.

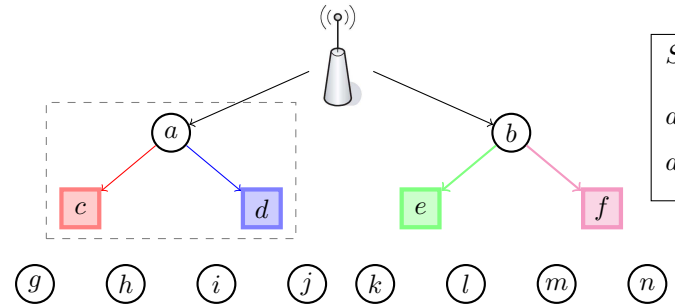
4.4.3 Détails du protocole

SMRP peut être facilement intégré à la version de base de SEIF. Pour cela, on a besoin seulement de modifier la phase de distribution de marques ; le reste des phases reste inchangé.

Dans la version basique, la SB distribue les marques directement aux racines. Dans SMRP, ces marques seront destinées aux sous-racines, tandis que les nœuds racines serviront seulement comme *passerelles*. Afin d'éviter des retransmissions inutiles, la SB choisit un sous-ensemble des nœuds racines couvrant toutes les sous-racines actives. Cela peut se faire en construisant un ensemble dominant DS à partir des voisins directs couvrant tous les nœuds à deux sauts. Pour chaque nœud $i \in DS$, la SB envoie un ensemble de marques de chaînes distinctes destinées aux sous-racines :

$$\begin{aligned} SB \rightarrow i : \quad & E(K_i, subRoot_1 || \ n_1 \ || \ P_1 \ || \ V_1 \\ & || \ ... \ || \\ & subRoot_m || \ n_m \ || \ P_m \ || V_m \ || \ r) \end{aligned} \tag{4.4}$$

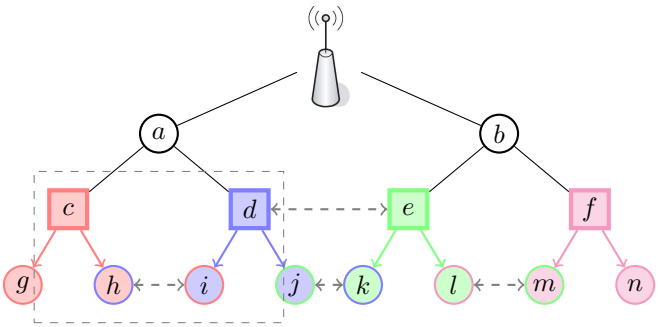
[3] Cela ne représente pas une grande exigence vu que les racines sont proches de la SB et que leur nombre est limité



$SB \rightarrow a : E(K_a, c||n_1||V_1||P_1||d||n_2||V_2||P_2||r)$

$a \rightarrow c : E(K_{a,c}, n_1||V_1||P_1||r)$
$a \rightarrow d : E(K_{a,d}, n_2||V_2||P_2||r)$

(a) En combinant SEIF et SMRP, le nœud racine a devient seulement un relais. Les marques V_1 et V_2 sont destinées aux sous-branches c et d.



$c \rightarrow * : c, n_1, V_1, P_1, r, NP_c$
$d \rightarrow * : d, n_2, V_2, P_2, r, NP_d$

$h \rightarrow * : h, n_1, V_1, P_1, r, NP_h$
$i \rightarrow * : i, n_2, V_2, P_2, r, NP_i$

(b) Le reste des nœuds exécutent exactement la même procédure que dans la première version de SEIF. On remarque qu'on peut découvrir plus de chemins en tolérant l'existante de chemins à l'intérieur d'une même branche.

Fig. 4.3: Un exemple d'exécution de SEIF avec l'extension SMRP

où :
- $subRoot_k$ représente l'ID d'une des sous-racines couvertes par i.
- n_k est l'ID de la chaîne affectée à $subRoot_k$ durant le tour courant.
- V_k est la première valeur non utilisée de la chaîne n_k.
- P_k est la position de V_k au sein de sa chaîne.
- m est le nombre de sous-racines couvertes par i.
- r représente le numéro de séquence du tour courant prélevé de la chaîne R.

Quand la racine i reçoit le message (4.4), elle vérifie que $RV_i = F(r)$. Si cette condition est vérifiée, le vérificateur RV_i est mis à jour. Par la suite, i authentifie chaque valeur V_k reçue en utilisant ces deux relations :

$$\begin{cases} P_k > P_{i,n_k} \\ BV_{i,n_k} = F^{P_k - P_{i,n_k}}(V_k) \end{cases}$$

Les variables P_{i,n_k} et BV_{i,n_k} sont mises à jour en conséquence. La phase finale durant la distribution des marques est le relais de chaque marque à la sous-racine cible en utilisant le message suivant :

$$i \rightarrow subRoot_k : E(K_{i,subRoot_k}, n_k \mid\mid P_k \mid\mid V_k \mid\mid r) \tag{4.5}$$

Lorsque la sous-racine reçoit ce message, elle effectue exactement les mêmes procédures des racines dans la version basique de SEIF, décrites dans le paragraphe. Le reste du réseau se comporte aussi de la même manière.

4.5 Analyses et simulations

Afin d'évaluer les performances de notre solution, nous avons implémenté le protocole SEIF en utilisant l'environnement TinyOS – version 1.15. TinyOS offre un environnement complet de développement pour les réseaux de capteur sans fil. Il inclut un système d'exploitation open-source conçu pour les systèmes embarqués, plus particulièrement pour les WSN. Ce système est devenu de nos jours la solution *de facto* du marché, et presque toutes les firmes de production de motes testent leurs plateformes avec ce système d'exploitation.

Cet environnement inclut aussi un simulateur, nommé Tossim [30], qui permet de tester une application TinyOS au sein d'un réseau sans fil virtuel sur des machines x86. De cette façon, le développeur peut gagner un temps précieux lors du debuggage d'applications, puisque que le test de l'application sur des motes réels au sein d'un réseau multi-hops n'est pas toujours évident. Il faut noter aussi que Tossim fournit un avantage très précieux par rapport aux autres simulateurs (tels que NS-2, OmNet et GloMoSim) : le développeur n'a pas besoin de réécrire l'application pour l'évaluer dans un réseau réel, car un même code TinyOS peut être exécuté sur de vrais motes ou bien au sein du simulateur Tossim.

Néanmoins, Tossim ne permet pas d'évaluer avec exactitude la consommation énergétique des nœuds. Pour cela, il existe d'autres outils permettant de simuler un réseau TinyOS et de fournir des rapports détaillés sur la consommation d'énergie des différents composants d'un mote : CPU, radio, ... *etc.* Nous avions à choisir entre deux alternatives possibles : PowerTossim et Avrora.

PowerTossim [37] est une extension du simulateur Tossim qui rajoute plus de précision lors de l'évaluation de la consommation d'énergie. Pour cela, PowerTossim effectue une double compilation : une première compilation est effectuée pour l'architecture Avr afin d'obtenir un profil d'exécution sur les motes réels. La deuxième compilation est effectuée pour l'architecture x86 pour produire le programme de simulation. En combinant l'information du profil de la première compilation avec la trace d'exécution de la seconde compilation, PowerTossim permet d'avoir des résultats plus proches de la réalité sur la consommation d'énergie.

Avrora [42] utilise une approche plus intéressante. Au lieu d'utiliser un binaire x86 pour simuler l'application – ce qui cache beaucoup de détails d'exécution – Avrora *émule* un micro-contrôleur Avr capable d'exécuter une application TinyOS au niveau machine. Par conséquent, Avrora peut donner une trace d'exécution complète de l'application dans un environnement

semblable aux motes réels, ce qui permet d'avoir une meilleure précision lors de l'évaluation de la consommation de l'énergie.

Afin de comparer les performances avec les solutions existantes, nous avons aussi implémenté les protocoles les plus représentatifs de la littérature [4] :
- SeRINS
- EINSENS
- SecRout
- TinyOS Beaconing

Le dernier protocole, TinyOS Beaconing, est en fait un protocole non sécurisé. Il a été choisi afin de donner une estimation sur une consommation optimale. En effet, ce protocole construit un arbre d'une manière très basique, en n'utilisant qu'un message par capteur. Cela représente donc la borne minimale de la consommation d'énergie afin d'assurer une connexité minimale au sein du réseau.

Toutes les opérations cryptographiques ont été programmées à l'aide de la librairie TinySec [17]. Cette librairie offre une solution complète de sécurité au niveau de la couche lien, mais on a désactivé cette fonctionnalité et on n'a utilisé que les primitives cryptographiques disponibles, tels que le chiffrement et le hachage. Pour cela, TinySec offrent ces deux opérations sous deux versions : MD5 et Skipjack. Ce dernier a été employé par notre implémentation pour effectuer les opérations de chiffrement et de hachage.

4.5.1 Consommation d'énergie

En utilisant l'outil Avrora, nous avons évalué la consommation moyenne d'un capteur durant un tour de construction des routes. Comme Avrora effectue une émulation de l'exécution du code au niveau instruction machine, tous les détails d'exécution sont évalués, y compris les opérations cryptographiques.

La Fig. 4.4 montre la variation de la consommation d'énergie par rapport à la taille du réseau. La conclusion principale déduite de ces simulations est que les solutions MANET point-à-point, basées sur un paradigme RREQ-RREP, sont inappropriés pour des applications de collection de données avec un WSN. Cette lacune de scalabilité est principalement causée par la découverte répétitive et non-corrélée, puisque chaque nœud débute une nouvelle découverte sans coordination avec les autres nœuds. Une découverte étant une diffusion générale dans tout le réseau, cette procédure devient trop coûteuse quand elle est répétée par chaque capteur du réseau.

Ces simulations permettent aussi de conclure que l'overhead généré par les opérations cryptographiques, et les informations secrètes échangées par radio, ne sont pas significatifs, comme le montre les performances de SEIF et EINSENS par rapport à TinyOS Beaconing.

[4] Le protocole INSENS a été implémenté mais nous ne pouvions pas obtenir de résultats pour des réseaux assez grands, vu que ce protocole génère un nombre très important de messages et d'opérations cryptographiques.

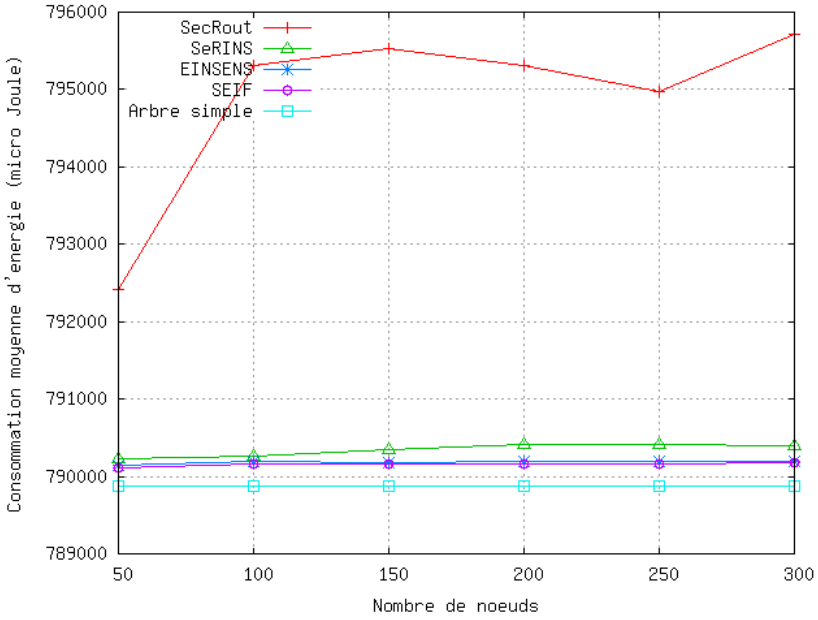


Fig. 4.4: Variation de la consommation d'énergie moyenne par rapport à la taille du réseau

4.5.2 Coût de détection

L'impact de la détection d'une attaque représente un autre critère de comparaison des solutions existantes. Le coût engendré lors d'une attaque est l'ensemble des ressources employées afin de détecter et bloquer ces comportements malicieux. Notons qu'on s'intéresse ici aux attaques contre la phase de construction des routes, qui essaient d'injecter de fausses informations de contrôle.

Pour les protocoles distribués, tels que SEIF et EINSENS, ce coût est très négligeable, vu que les attaques sont automatiquement stoppées par les nœuds eux-mêmes, et cela sans collaboration ni messages échangés. Ainsi, chaque nœud se base sur son information locale, qui lui suffit pour détecter ces attaques avec succès. Le seul coût engendré est le coût de calcul afin de vérifier l'authenticité des informations de routage reçues. Mais comme les opérations nécessaires sont assez simples – des fonctions de hash pour la plus part des cas – ce coût n'a pas un grand impact.

En revanche, SeRINS est un protocole hybride dans lequel les capteurs ne peuvent effectuer que des vérifications partielles limitant leur capacité à prendre des décisions locales en présence de messages suspects. Dans ce genre de circonstances, les nœuds détecteurs doivent contacter la SB avec des diffusions générales, ce qui pénalise le réseau en consommation d'énergie.

Notons que l'approche centralisée d'INSENS lui permet d'avoir un overhead très réduit. En effet, les fausses informations de routage n'ont aucune conséquence sur les nœuds, car ces derniers ne sont plus responsables des vérifications de sécurité, qui sont toutes effectuées au niveau de la SB.

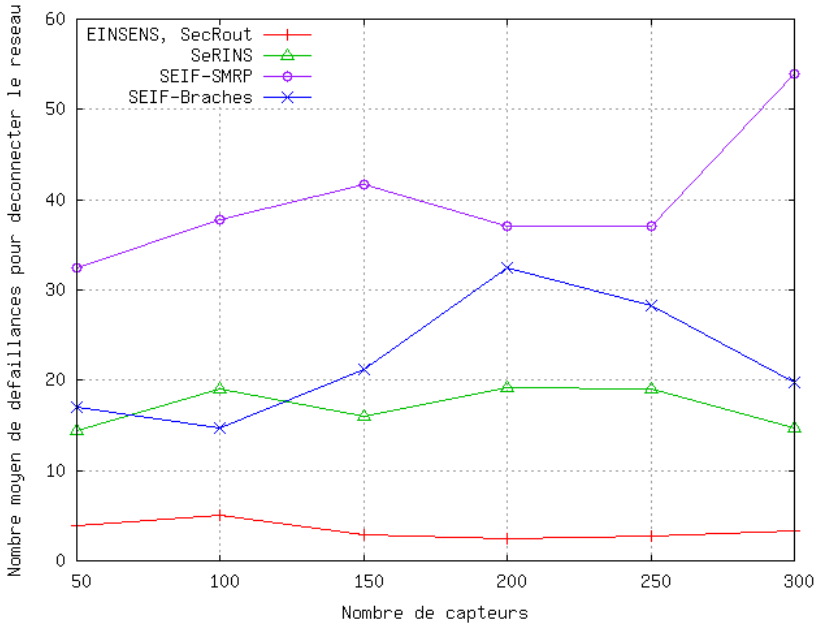


Fig. 4.5: Variation de la tolérance de la topologie par rapport à la taille du réseau

4.5.3 Tolérance

Dans plusieurs situations, les motes doivent être déployés dans des environnements hostiles et parfois inaccessibles, rendant le monitoring manuel assez pénible. Dans ce genre de situations, la durée de vie du réseau – qui est la période pendant laquelle le système est *fonctionnel* sans intervention extérieure – devient un critère primordial et d'une importance cruciale. Dans un contexte de routage, plusieurs définitions du terme « fonctionnel » peuvent être considérées. Dans notre étude, nous avons considéré qu'un système est fonctionnel lorsque chaque capteur actif du réseau peut faire parvenir ses données à la SB. Lorsqu'un capteur ne peut plus communiquer avec la SB à cause de la rupture de tous ses chemins, la topologie est considérée défaillante et une reconstruction est nécessaire.

Afin d'évaluer la résistance des solutions existantes, on a effectué des simulations permettant d'obtenir le nombre moyen de défaillances nécessaire pour *casser* une topologie en la rendant non-connexe. Cela se produit lorsqu'il existe certains nœuds du réseau n'ayant aucune route valide vers la SB. La Fig. 4.5 donne les résultats de simulation obtenus.

Les protocoles uni-chemin, tels que EINSENS et SecRout, fournissent la résilience la plus faible vu qu'ils n'offrent aucune redondance permettant de pallier les déconnexions des chemins. Les approches multi-chemins permettent d'obtenir de meilleurs résultats, mais avec des niveaux différents de fiabilité. SeRINS construit des chemins alternatifs sans considérer leur intersection, ce qui diminue le taux de duplication des données transférées via les chemins alternatifs. Le protocole SEIF, avec son extension des sous-branches disjointes, bénéficie d'un taux de duplication largement plus optimal, ainsi qu'une abondance de chemins alternatifs supérieure à celle des chemins disjoints classiques. La combinaison de ces deux paramètres

permet d'obtenir la plus grande tolérance contre les défaillances matérielles et les attaques d'intrus.

4.5.4 Overhead cryptographique

Lors de la conception d'un protocole sécurisé, il est nécessaire d'évaluer le coût engendré par les mécanismes cryptographiques utilisés. Ce coût inclut plusieurs paramètres, dont le plus important est l'overhead radio. Ce dernier représente le nombre de bits nécessaires pour communiquer les informations de contrôle et leurs protections entre les nœuds voisins. Ce critère est très important, car certaines études ont montré qu'un bit envoyé par radio est équivalent, en énergie, à l'exécution de 800-1000 instructions machines.

	Format	**Taille (bits)**
EINSENS	$E(src, MAC)$	80
SecRout	$src, dst, id, nonce, MAC, this, pre$	144
SEIF	$src, branch, 3 \times MAC$	224
SeRINS	$src, hops, 5 \times MAC$	344

Tab. 4.1: Taille des messages RREQ des protocoles sécurisés implémentés

Le tableau 4.1 résume le format des messages utilisés par chaque protocole, ainsi que la taille résultante. Pour cela, on a choisi de comparer seulement les messages RREQ, car ils représentent les messages utilisés durant la phase de construction, qui se réitère à chaque tour. Les messages d'initialisation de certains protocoles ont été omis, vu que l'opération d'initialisation ne s'effectue qu'une seule fois, et peut ne pas toucher tout le réseau.

Ces chiffres obtenus sont basés sur les hypothèses suivantes :
- Une adresse logique d'un nœud occupe 16 bits.
- Un MAC occupe 64 bits.
- Une opération de chiffrement résulte en un nombre entier d'octet, signifiant qu'il y a un éventuel padding par des 0.
- Les nonces et autres informations secondaires occupent 8 bits.

4.6 Conclusion

On a présenté dans ce chapitre notre solution de dissémination fiable, appelée SEIF. Ce protocole est une solution sécurisée de construction d'une topologie de routage multi-chemins, offrant une grande tolérance aux intrus et aux pannes. À l'aide de son fonctionnement totalement distribué, les nœuds se basent seulement sur des mécanismes localisés afin de détecter les éventuelles attaques, rendant la détection rapide et peu gourmande en ressources. En plus,

grâce au mécanisme SMRP, notre solution tolère mieux les pannes et augmente significativement la durée de fonctionnement du réseau.

Ce chapitre a décrit aussi les résultats de simulation obtenus grâce à une implémentation de SEIF avec TinyOS. Afin de comparer ces résultats avec les performances d'autres solutions, nous avons implémenté quatre autres protocoles existants : SeRINS, SecRout, EINSENS et TinyOS Beaconing. Les performances affichées confirment notre conception.

Conclusion Générale

L'informatique embarquée sans fil connaît actuellement un essor remarquable. Parmi les différentes familles de cette technologie, les réseaux de capteurs sans fil ont su attirer récemment l'attention des industriels grâce à leur simplicité et leur utilité dans divers domaines d'activité de la société. Le challenge principal de ce nouveau paradigme est de trouver des solutions fiables pouvant faire fonctionner ce système sous les différentes contraintes matérielles et logicielles.

Dans ce travail, on s'est intéressé au problème de fiabilité de dissémination dans les réseaux de capteurs sans fil. Ce problème se trouve au cœur de la question de fiabilité, vu la position cruciale qu'occupe la couche de routage au sein de l'architecture de communication. Nous avons présenté les principales approches existantes, leurs avantages et leurs inconvénients. Cela nous a permis de proposer notre solution, nommée SEIF, offrant une nouvelle forme de construction distribuée et sécurisée d'une topologie multi-path adaptée au paradigme plusieurs-à-un des réseaux de capteurs. SEIF fait partie donc des protocoles tolérants aux pannes et aux intrus. Mais contrairement aux solutions existantes, SEIF assure un compromis entre le niveau de la tolérance et la consommation en énergie. En effet, SEIF propose une amélioration des chemins disjoints en tolérant une certaine intersection contrôlée entre les routes découvertes. Cela permet d'augmenter considérablement la tolérance de la topologie sans rajouter de nouveaux messages de contrôles. Des simulations avec Tossim ont permis de démontrer cet apport.

La version étendue de SEIF étudiée dans ce travail se limite à placer les sous-racines à deux sauts de la station de base. Cela signifie qu'on tolère l'existence d'un seul nœud en intersection entre les chemins découverts. Cependant, il serait intéressant de voir l'impact de rajouter plusieurs nœuds en intersection au début du chemin sur le niveau de tolérance offert. À priori, on peut dire que ce taux d'intersection peut dépendre de la taille du réseau et sa densité. En effet, en tolérant trop de nœuds en intersection dans des réseaux à petite taille, on se retrouverait avec une topologie qui tend vers les chemins uni-paths. Dans des réseaux plus grands, le fait de limiter l'intersection à un seul niveau sera semblable au problème de

chemins disjoints dans le mécanisme de branches initial. Par conséquent, il serait intéressant de trouver la valeur optimale du taux d'intersection en considérant les différents scénarios de déploiement et les propriétés des graphes résultants.

Un autre problème intéressant est le degré optimal des nœuds. Lorsqu'un nœud découvre un certain nombre de chemins alternatifs, il pourra les utiliser pour transmettre ses données, afin d'éviter les éventuelles pannes lors du relais. Plus le nombre de chemins augmente, plus la probabilité de perte diminue. En contre partie, plus le nombre de chemins augmente, plus la consommation globale en énergie augmente, car on sera obligé de faire coopérer plus de nœuds pour router l'information. Par conséquent, étudier le nombre optimal de chemins à utiliser pour le routage, en considérant l'énergie totale du réseau et sa durée de vie, se révèle un problème intéressant.

Bibliographie

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks : A survey. *Computer Networks*, pages 393 – 422, 2002.

[2] S.P. Beeby, R.N. Torah, M.J. Tudor, P. Glynne-Jones, T. O'Donnell, C.R. Saha, and S. Roy. A micro electromagnetic generator for vibration energy harvesting. *Journal of Micromechanics and Microengineering*, 17(7) :1257–1265, July 2007.

[3] Computer Security Institute. CSI Survey 2007 : The 12th Annual Computer Crime and Security Survey. http ://www.gocsi.com.

[4] N. Cooprider, W. Archer, E. Eide, D. Gay, and J. Regehr. Efficient memory safety for tinyos. In *SenSys '07 : Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 205–218, New York, NY, USA, 2007. ACM.

[5] J. Deng, R. Han, and S. Mishra. INSENS : Intrusion-tolerant routing in wireless sensor networks. *Technical Report CU CS-939-02, Department of Computer Science, University of Colorado*, 2002.

[6] J. Deng, R. Han, and S. Mishra. INSENS : Intrusion-tolerant routing for wireless sensor networks. *Computer Communications*, 29(2) :216–230, 2006.

[7] A. Dunkels, B. Gronvall, and T. Voigt. Contiki : a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the 29th annual IEEE international conference on Local Computer Networks (LCN 04)*, pages 455–462, Washington, DC, USA, 2004. IEEE Computer Society.

[8] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He. Software-based on-line energy estimation for sensor nodes. In *Proceedings of the 4th workshop on Embedded networked sensors (EmNets 07)*, pages 28–32, New York, NY, USA, 2007. ACM.

[9] A. Dunkels, O. Schmidt, T. Voigt, and M. Ali. Protothreads : simplifying event-driven programming of memory-constrained embedded systems. In *Proceedings of the 4th international conference on Embedded networked sensor systems (SenSys 06)*, pages 29–42, New York, NY, USA, 2006. ACM.

[10] A. El-Hoiydi and J.-D. Decotignie. Wisemac : an ultra low power mac protocol for the downlink of infrastructure wireless sensor networks. In *Proceedings of the Ninth International Symposium on Computers and Communications 2004 Volume 2 (ISCC 04)*, pages 244–251, Washington, DC, USA, 2004. IEEE Computer Society.

[11] J. C. Foster. *Buffer Overflow Attacks*. Syngress Publishing, 2005.

[12] M. Guerrero Zapata. Secure ad hoc on-demand distance vector (saodv) routing, September 2006. INTERNET-DRAFT draft-guerrero-manet-saodv-06.txt.

[13] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister. System architecture directions for networked sensors. In *Proceedings of Architectural Support for Programming Languages and Operating Systems*, pages 93 – 104, 2000.

[14] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. *SenSys 04*, pages 81–94, 2004.

[15] J. Jeong and D. Culler. Incremental network programming for wireless sensors. *IEEE SECON 04*, pages 25–33, 2004.

[16] X. Jiang, P. Dutta, D. Culler, and I. Stoica. Micro power meter for energy monitoring of wireless sensor networks at scale. In *Proceedings of the 6th international conference on Information processing in sensor networks (IPSN 07)*, pages 186–195, New York, NY, USA, 2007. ACM.

[17] C. Karlof, N. Sastry, and D. Wagner. TinySec : a link layer security architecture for wireless sensor networks. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SensSys 2004)*, November 2004.

[18] C. Karlof and D. Wagner. Secure Routing in Wireless Sensor Networks : Attacks and Countermeasures. *Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols*, 1(2–3) :293–315, September 2003.

[19] K. Kredo and P. Mohapatra. Medium access control in wireless sensor networks. *Comput. Netw.*, 51(4) :961–994, 2007.

[20] L. Lamport. Constructing digital signatures from one-way function. *Technical Report SRI-CSL-98, SRI International*, 1979.

[21] S. B. Lee and Y. H. Choi. ARMS : An Authenticated Routing Message in Sensor Networks. In LNCS 4074, editor, *Secure Mobile Ad-hoc Networks and Sensors Workshop (MADNES 05)*, pages 158 – 173, 2005.

[22] S. B. Lee and Y. H. Choi. A secure alternate path routing in sensor networks. *Computer Communications*, 30(1) :153—165, December 2006.

[23] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer, and D. Culler. The emergence of networking abstractions and techniques in tinyos, 2004.

[24] A. Liu and P. Ning. TinyECC : a configurable library for elliptic curve cryptography in wireless sensor networks. In *Proceedings of the 2008 International Conference on Information Processing in Sensor Networks (ISPN 2008)*, pages 245–256, Washington, DC, USA, 2008. IEEE Computer Society.

[25] W. Lou and Y. Kwon. H-SPREAD : a Hybrid Multipath Scheme for Secure and Reliable Data Collection in Wireless Sensor Networks. *IEEE Transactions on Vehicular Technology*, 55(4) :1320–1330, 2006.

[26] G. Mao, B. Fidan, and B. Anderson. Wireless sensor network localization techniques. *Comput. Netw.*, 51(10) :2529–2553, 2007.

[27] A. Ouadjaout, Y. Challal, N. Lasla, and M. Bagaa. SEIF : Secure and Efficient Intrusion-Fault Tolerant Routing Protocol for Wireless Sensor Networks. In IEEE Computer Society, editor, *proceedings of the Third International Conference on Availability, Reliability and Security (ARES 08)*, pages 503–508, March 2008.

[28] J. Panchard, S. Rao, T. V. Prabhakar, J.-P. Hubaux, and H. S. Jamadagni. COMMONSense Net : A Wireless Sensor Network for Resource-Poor Agriculture in the Semiarid Areas of Developing Countries. *Information Technologies and International Development*, 4(1) :51–67, 2007.

[29] C. Park and K. Lahiri. Battery discharge characteristics of wireless sensor nodes : An experimental analysis. In *Proceedings of the IEEE Conf. on Sensor and Ad-hoc Communications and Networks (SECON)*, 2005.

[30] L. Philip, L. Nelson, W. Matt, and C. David. TOSSIM : Accurate and scalable simulation of entire tinyos applications. In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, 2003.

[31] K. Piotrowski, P. Langendoerfer, and S. Peter. How public key cryptography influences wireless sensor node lifetime. In *Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks (SASN 06)*, pages 169–176, New York, NY, USA, 2006. ACM.

[32] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys 04)*, pages 95–107, New York, NY, USA, 2004. ACM Press.

[33] C. Qing, A. Tarek, J. Stankovic, and H. Tian. The LiteOS Operating System : Towards Unix-like Abstractions for Wireless Sensor Networks. In *Proceedings of the 2008 International Conference on Information Processing in Sensor Networks (ISPN 2008)*, pages 233–244, Washington, DC, USA, 2008. IEEE Computer Society.

[34] S. R. Madden et al. The Design of an Acquisitional Query Processor for Sensor Networks. In *SIGMOD*, June 2003.

[35] Sensys Networks. http ://www.sensysnetworks.com.

[36] S. C. Seo, D .G. Han, H. C. Kim, and S. Hong. TinyECCK : Efficient Elliptic Curve Cryptography Implementation over GF(2m) on 8-Bit Micaz Mote. *IEICE Transactions on Information and Systems*, pages 1338–1347, 2008.

[37] V. Shnayder, M. Hempstead, B. Chen, G. Werner Allen, and M. Welsh. Simulating the power consumption of large-scale sensor network applications. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pages 188–200, New York, NY, USA, 2004. ACM Press.

[38] T. Stathopoulos, J. Heidemann, and D. Estrin. A remote code update mechanism for wireless sensor networks. *Technical Report CENS Technical Report 30*, 2003.

[39] H. Tan, S. Jha, D. Ostry, J. Zic, and V. Sivaraman. Secure multi-hop network programming with multiple one-way key chains. In *WiSec '08 : Proceedings of the first ACM conference on Wireless network security*, pages 183–193, New York, NY, USA, 2008. ACM.

[40] Technology Review. *10 Emerging Technologies That Will Change the World*, February 2003.

[41] The Institute of Electrical and Electronics Engineers. *Part 15.4 : Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*, 2006.

[42] B. L. Titzer, D. K. Lee, and J. Palsberg. Avrora : scalable sensor network simulation with precise timing. In *Proceedings of the 4th International Symposium on Information Processing in sensor Networks (IPSN)*, page 67, Piscataway, NJ, USA, 2005.

[43] R. Von Behren, J. Condit, and E. Brewer. Why events are a bad idea (for high-concurrency servers). In *Proceedings of the 9th conference on Hot Topics in Operating Systems (HOTOS 03)*, pages 4–4, Berkeley, CA, USA, 2003. USENIX Association.

[44] L. Wang. MNP : multihop network reprogramming service for sensor networks. *SenSys '04*, pages 285–286, 2004.

[45] R. Xiuli and Y. Haibin. A novel multipath disjoint routing to support ad hoc wireless sensor networks. In *Proceedings of the Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC 06)*, pages 174–178, Washington, DC, USA, 2006.

[46] J. Yin and S. Madria. Secrout : A secure routing protocol for sensor networks. In *AINA 06 : Proceedings of the 20th International Conference on Advanced Information Networking and Applications*, volume 1, pages 393–398, Washington, DC, USA, 2006. IEEE Computer Society.

[47] M. Z. Zamalloa and B. Krishnamachari. An Analysis of Unreliability and Asymmetry in Low-Power Wireless Links. *ACM Transactions on Sensor Networks*, 3(2), 2007.

[48] S. Zhu, S. Setia, and S. Jajodia. LEAP : Efficient security mechanisms for large-scale distributed sensor networks. In *Proceedings of ACM CCS*, 2003.

[49] S. Zhu, S. Setia, and S. Jajodia. LEAP+ : Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks. *ACM Transactions on Sensor Networks*, 2(4) :500–528, November 2006.