Modèles et mécanismes pour la protection contre les attaques par déni de service dans les réseaux de capteurs sans fil

Quentin Monnet

Université Paris-Est

École doctorale MSTIC

Thèse de Doctorat

pour obtenir le titre de

Docteur d'Université Paris-Est

Spécialité : Informatique

défendue par

Quentin Monnet

Modèles et mécanismes pour la protection contre les attaques par déni de service dans les réseaux de capteurs sans fil

Directrice de thèse : Prof. Lynda Mokdad

soutenue le 17 juillet 2015 devant le jury composé de :

Directrice :

Prof. Lynda Mokdad ..... Université Paris-Est Créteil

Rapporteurs :

Prof. Madjid Merabti ..... Liverpool John Moores University, Royaume-Uni

Prof. Samir Tohmé ..... Université de Versailles-St-Quentin-en-Yvelines

Examinateurs :

Dr Paolo Ballarini ..... École centrale Paris

Prof. Jalel Ben-Othman ..... Université Paris 13

Prof. Karim Djouani ..... Université Paris-Est Créteil

Prof. Jean-Michel Fourneau ... Université de Versailles-St-Quentin-en-Yvelines

Prof. Véronique Vèque ..... Université Paris-Sud

UNIVERSITÉ PARIS-EST

Thèse préparée au LACL
(Laboratoire d'Algorithmique, Complexité et Logique)
https://lacl.fr

LACL
Faculté des Sciences et Technologie
61 avenue du Général De Gaulle
94 010 Créteil

Résumé

Modèles et mécanismes pour la protection contre les attaques par déni de service dans les réseaux de capteurs sans fil

Composés d'appareils fortement limités en ressources (puissance de calcul, mémoire et énergie disponible) et qui communiquent par voie hertzienne, les réseaux de capteurs sans fil composent avec leurs faibles capacités pour déployer une architecture de communication de manière autonome, collecter des données sur leur environnement et les faire remonter jusqu'à l'utilisateur. Des « transports intelligents » à la surveillance du taux de pollution environnemental, en passant par la détection d'incendies ou encore l'« Internet des objets », ces réseaux sont aujourd'hui utilisés dans une multitude d'applications. Certaines d'entre elles, de nature médicale ou militaire par exemple, ont de fortes exigences en matière de sécurité.

Les travaux de cette thèse se concentrent sur la protection contre les attaques dites par « déni de service », qui visent à perturber le fonctionnement normal du réseau. Ils sont basés sur l'utilisation de capteurs de surveillance, qui sont périodiquement renouvelés pour répartir la consommation en énergie. De nouveaux mécanismes sont introduits pour établir un processus de sélection efficace de ces capteurs, en optimisant la simplicité de déploiement (sélection aléatoire), la répartition de la charge énergétique (sélection selon l'énergie résiduelle) ou encore la sécurité du réseau (élection démocratique basée sur un score de réputation). Sont également fournis différents outils pour modéliser les systèmes obtenus sous forme de chaines de Markov à temps continu, de réseaux de Petri stochastiques (réutilisables pour des opérations de model checking) ou encore de jeux quantitatifs.

Mots-clés : Réseaux de capteurs sans fils • énergie • sécurité • déni de service • détection d'intrusion • simulation • modélisation • théorie des jeux.

Abstract

Mechanisms and Modeling Tools for Protection Against Denial of Service Attacks in Wireless Sensor Networks

Wireless sensor networks are made of small devices with low resources (low computing power, little memory and little energy available), communicating through electromagnetic transmissions. In spite of these limitations, sensors are able to self-deploy and to auto-organize into a network collecting, gathering and forwarding data about their environment to the user. Today those networks are used for many purposes: "intelligent transportation", monitoring pollution level in the environment, detecting fires, or the "Internet of things" are some example applications involving sensors. Some of them, such as applications from medical or military domains, have strong security requirements.

The work of this thesis focuses on protection against "denial of service" attacks which are meant to harm the good functioning of the network. It relies on the use of monitoring sensors: these sentinels are periodically renewed so as to better balance the energy consumption. New mechanisms are introduced so as to establish an efficient selection process for those sensors: the first one favors the ease of deployment (random selection), while the second one promotes load balancing (selection based on residual energy) and the last one is about better security (democratic election based on reputation scores). Furthermore, some tools are provided to model the system as continuous-time Markov chains, as stochastic Petri networks (which are reusable for model checking operations) or even as quantitative games.

***Keywords**: Wireless Sensor Networks • Energy • Security • Denial of Service • Intrusion Detection • Simulation • Modeling • Game Theory.*

Remerciements

Décrire le doctorat comme la seule préparation d'un diplôme serait une injustice criante. Il y aurait beaucoup à en dire, mais j'ai choisi, avant de plonger dans la rigueur des sciences informatiques, d'en retenir un point de vue spécifique : le doctorat est un ensemble de rencontres.

Il y a d'abord les rencontres en amont. Lynda m'a eu comme élève en Master, et je lui dois beaucoup de cette thèse : sujet, financement, encadrement, débats et coopération. Du début à la fin, elle m'a guidé et surtout m'a fait confiance, et je l'en remercie profondément.

Pendant la thèse, il y a ceux que j'ai rencontrés en arrivant, parce qu'ils étaient au LACL avant moi ou bien parce qu'ils sont arrivés tout en même temps. Les doctorants sont sans doute ceux avec qui j'ai le plus échangé : Dimitrios, Louis, Mohamed, Muath m'ont précédé, Yoann m'a suivi de bureau en bureau, de théorie fumeuse en complot pour refaire le monde, machine après machine. C'est tout naturellement que je les remercie, de même que je remercie l'ensemble de l'équipe du laboratoire, pour leur accueil et leurs nombreux conseils.

Et il y a toutes les rencontres effectuées pendant, au cours de, durant cette thèse ; pour travailler, pour enseigner, par chance, par hasard. La recherche m'a permis de travailler avec plusieurs personnes : merci à Jalel, à Mathieu et à Paolo, pour ne citer qu'eux. À Créteil comme à Dauphine, j'ai découvert et pris gout à l'enseignement ; j'ai certes croisé beaucoup d'élèves, mais aussi rencontré des collègues avec qui l'organisation des cours est un plaisir. Merci notamment à Antoine, Julien, Luidnel et Pascal pour les cours de systèmes d'exploitation. Et pour le Beaufort.

Toujours pendant la thèse, et au fil des ans, d'autres individus, plus ou moins recommandables, ont rejoint les rangs des doctorants du LACL ; à défaut d'avoir réussi à leur imposer la supériorité de ma condition d'« ancien », je m'en suis fait des amis. Merci à Aurélien, Martin, Nghi, Raouf, Rodica, Sergiu, Thomas et Victor pour les repas et les discussions, les logiciels libres, les dessins étranges, les métalangages et les comorphismes. Entre autres choses.

This Ph. D. brought me to Liverpool for two months, and I was lucky enough to meet wonderful people there. I have to thank Prof Merabti and his team for their hospitality at LJMU. Also I am grateful to Aine, Brett, David, Mark, Rob, Will and the many other Ph. D. students there from whom I received a more-than-warm welcome.

Il y a les rencontres futures, qui je l'espère seront nombreuses, et sauront profiter à tous. Mais il m'est difficile de leur dédier des remerciements aujourd'hui…

Enfin il est de ces rencontres qui sont hors contexte, intemporelles, qui ne sont ni à l'instant ni au lieu de la thèse. Ma famille et mes proches, qui m'ont soutenu, supporté parfois pendant ces années, se retrouvent lésés : car je ne pourrai jamais les remercier assez.

Table des matières

Table des figures

Liste des tableaux

1

Introduction

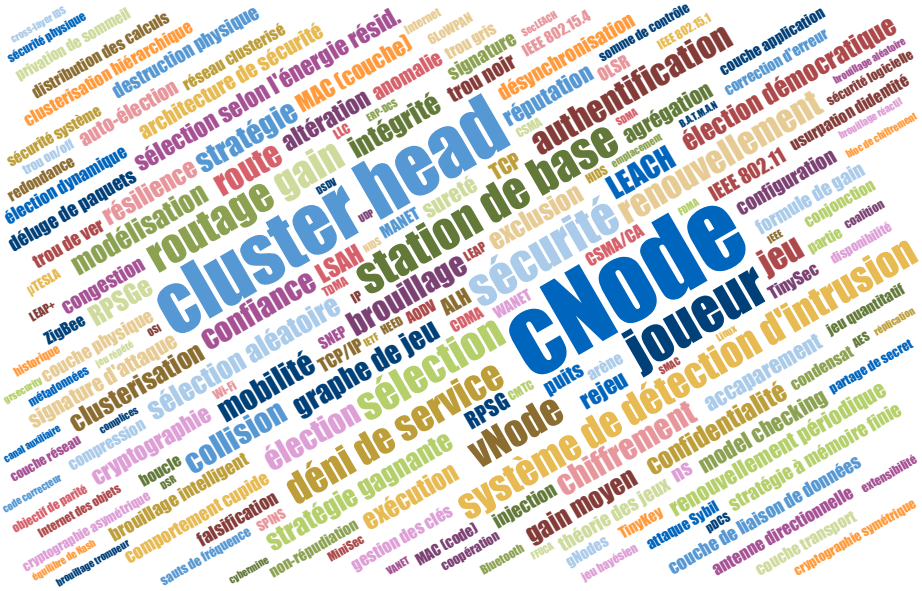

Fig. 1.1 : Nuage des mots-clés indexés de l'ouvrage. L'échelle des mots est relative à la racine carrée du nombre d'occurrences.

1 Réseaux de capteurs sans fil et déni de service

La lutte contre les attaques de type « déni de service » dans les réseaux de capteurs sans fil est le fil directeur des travaux présentés dans cet ouvrage. Les réseaux de capteurs sont constitués de petits appareils — les capteurs — équipés d'un module de communication sans fil. Dispersés dans l'environnement à étudier, ces capteurs sont chargés de réaliser des mesures physiques, de les convertir en un signal numérique, et de les rapatrier pour un traitement plus approfondi à une station de base, qui fait office d'interface entre le réseau et l'utilisateur. Cette collecte d'information est soumises aux contraintes en ressources des capteurs, dont les capacités en calcul, en mémoire, et dont les réserves d'énergie disponible (sous forme de batteries) sont limitées.

Mais les protocoles déployés ont su être adaptés : les capteurs sont aujourd'hui utilisés dans une multitude d'applications dans des domaines aussi variés que l'environnement, la santé, l'urbanisme, les transports, la domotique, et bien sûr le domaine militaire. Liés à la fois aux avancées technologiques et à l'émergence de nouveaux usages dans l'exploitation des données, les concepts de « transports intelligents », de « villes intelligentes » ou d'« Internet des objets » se développent peu à peu et semblent promettre un usage de plus en plus intensif des réseaux de capteurs sans fil.

Toutefois, ces réseaux particuliers introduisent leur lot de problématiques : à côté des contraintes fortes en ressources se pose la question de la sécurité, dont la mise en place est une nécessité absolue pour les applications médicales ou militaires par exemple. Alors que les mécanismes d'authentification et de chiffrement font appel, dans la plupart des systèmes informatiques, à des protocoles cryptographiques couteux en ressources, il a fallu adapter les mécanismes au monde des capteurs. Mais la sécurité comporte d'autres volets, et la disponibilité des réseaux, suivant le contexte, peut s'avérer tout aussi essentielle.

En résumé, le problème se pose ainsi : comment prévenir, ou à défaut comment détecter puis contourner, tout en économisant les ressources des capteurs, une action d'origine malveillante qui viserait à mettre le réseau hors service ?

Cette thèse s'appuie sur l'attribution d'un rôle de surveillance à certains des capteurs, chargés de détecter des comportements hostiles au sein du réseau, et de prévenir leurs pairs lorsqu'une attaque est détectée.

C'est notamment le processus de sélection dynamique de ces capteurs de surveillance qui fait l'objet d'une étude poussée. Plusieurs solutions sont proposées : une sélection aléatoire, permettant d'obtenir statistiquement une bonne couverture du réseau ; une sélection basée sur l'énergie résiduelle des capteurs, dont l'avantage est d'offrir une meilleure répartition de la charge (en matière de consommation énergétique) dans le réseau ; et enfin un processus d'élection démocratique, basé sur des scores de réputation, qui améliore encore la sécurité du dispositif.

Mais en sus des algorithmes concrets, il est parfois utile de pouvoir représenter un processus sous un aspect plus formel. Validées par des simulations, certaines des méthodes proposées sont également modélisées à l'aide de chaines de Markov ou de réseaux de Petri particuliers. En fin d'étude, les jeux quantitatifs sont utilisés pour caractériser à plus haut niveau le système composé d'un capteur corrompu et des capteurs de surveillance.

Le but des travaux présentés dans cette thèse est donc de proposer un ensemble de méthodes de détection et de réaction efficaces aux attaques par déni de service, tout en économisant ou en répartissant au mieux la consommation énergétique des capteurs pour prolonger le plus possible la durée de fonctionnement du réseau. Les outils de modélisation fournis permettent à la fois de valider ces méthodes, de mieux les comprendre, et de servir de base, dans le futur, pour la conception de mécanismes toujours plus performants.

2 Organisation de la thèse

Chapitre 2 : Architecture et enjeux des réseaux de capteurs Les réseaux de capteurs reposent sur des appareils faibles en capacités de calcul, en mémoire, ainsi qu'en termes d'énergie disponible. Leur déploiement, leurs communications, leur organisation autonome, ainsi que les tâches qui leur sont assignées doivent composer avec ces limites. Ce chapitre s'attache donc à présenter l'architecture basique des capteurs et des réseaux qui en sont constitués, à donner quelques exemples d'applications, puis à introduire les problématiques principales liées à ces réseaux : contraintes en ressources, mobilité éventuelle, sureté et résilience, et clusterisation hiérarchique du réseau en forment les grands axes.

Chapitre 3 : Sécurité dans les réseaux de capteurs sans fil La sécurité informatique est transversale à l'ensemble des composantes de ce domaine. Pour les réseaux de capteurs, les mécanismes de sécurité sont soumis aux mêmes restrictions en ressources que les autres protocoles. Pour autant, la sécurité ne doit pas être négligée ; elle est même indispensable pour toutes les applications critiques. Dans ce chapitre sont développées les principales problématiques qu'elle regroupe : confidentialité et authentification sont parmi les plus importantes, et ont fait l'objet, dans les réseaux de capteurs, de propositions spécifiques. La résistance aux attaques par déni de service est également un aspect important, qui bien sûr est central dans cette thèse. Il est donc abordé en plusieurs temps : après la catégorisation puis la description des attaques majeures connues, les méthodes de prévention, de détection puis de réaction sont développées.

Chapitre 4 : Sélection aléatoire des *cNodes* Une méthode de détection des attaques par déni de service consiste à assigner à certains capteurs (appelés *cNodes*) un rôle de surveillance au sein du réseau. Combinée à une partition en clusters du réseau, elle permet d'observer efficacement le trafic pour repérer les comportements anormaux. Se pose alors la question du renouvellement du rôle de surveillance : changer de *cNodes* au cours du temps permet de répartir la consommation engendrée par cette tâche de surveillance, et par là de prolonger la durée de vie du réseau dans son ensemble.

Le processus de sélection peut être effectué simplement en choisissant un nombre aléatoire et en le comparant à un seuil de probabilité : c'est la méthode, simple à mettre en œuvre, utilisée dans ce chapitre. Les simulations permettent d'observer l'amélioration importante obtenue sur la répartition de la charge énergétique et valident l'emploi d'un renouvellement dynamique des *cNodes*.

Afin de pouvoir l'analyser de façon plus formelle, ce renouvellement est également modélisé sous forme de chaines de Markov à temps continu, mais cet outil a

ses limites ; pour les contourner, une seconde méthode formelle est employée : il s'agit des réseaux de Petri stochastiques généralisés étendus, qui permettent une représentation efficace du modèle. De plus, il est possible d'en dériver des formules de logique stochastique avec automates hybrides, et des exemples de ces formules sont fournis. Ils permettent à leur tour de réutiliser le modèle obtenu par le biais de techniques de *model checking* pour vérifier diverses propriétés sur le système.

Chapitre 5 : Sélection des *cNodes* selon l'énergie résiduelle Une deuxième façon de renouveler les *cNodes* consiste à considérer leur énergie résiduelle à l'instant où le processus de sélection est exécuté, de sorte que les capteurs possédant les plus grandes réserves d'énergies se voient attribuer la tâche la plus couteuse. Mais il ne faut pas oublier que dans un contexte de détection des attaques, le rôle de *cNode* est particulièrement intéressant à endosser pour un capteur compromis, puisqu'il peut l'aider à éviter de se faire repérer. S'il n'est pas possible d'empêcher un capteur de « mentir » sur la valeur de son énergie résiduelle, des méthodes sont appliquées en revanche pour détecter ces tentatives frauduleuses d'accès au rôle. Elles passent par la nomination de capteurs (les *vNodes*) chargés de vérifier l'évolution de la consommation des *cNodes* afin de détecter les tentatives de fraude. Un second mécanisme est également mis en place pour assurer la bonne couverture du réseau par les *cNodes*, menacée par l'aspect déterministe du processus de sélection basé sur l'énergie restante des capteurs. Des simulations sont une nouvelle fois utilisées pour mettre en avant le gain supplémentaire apporté à la répartition de la charge en énergie.

Chapitre 6 : Élection démocratique des *cNodes* Plutôt que de se concentrer uniquement sur la répartition de la charge énergétique, il est possible d'envisager une approche cherchant à maximiser la sécurité du réseau. La troisième méthode proposée pour la sélection des *cNodes* consiste à les faire élire par les capteurs voisins, en fonction d'un score de réputation établi au fil du temps, c'est-à-dire en fonction de la confiance accumulée par chaque capteur pour ses « bonnes » actions (ou du moins l'absence de « mauvaises » actions). Les nœuds « élus » sont placés dans une liste de candidats de confiance. Le score final peut être ajusté en fonction d'autres critères, en reprenant par exemple l'énergie résiduelle, de sorte que les candidats « idéaux » au regard de la somme des critères retenus se voient attribuer le rôle de *cNode*. Cette troisième méthode est à son tour évaluée, puis comparée à celles des deux chapitres précédents, à travers de nouvelles simulations. Cette comparaison permet d'établir, en fonction des performances recherchées (ou du critère à favoriser), quelles sont les meilleures solutions à déployer pour la sélection des *cNodes* dans un réseau de capteurs sans fil.

Chapitre 7 : Modélisation à l'aide de jeux quantitatifs Le dernier chapitre technique revient sur des aspects de modélisation, cette fois-ci en se basant sur la théorie des jeux. La représentation ne porte plus sur le processus de sélection pour la surveillance, mais sur les interactions entre *cNodes* et capteurs compromis. Le système est modélisé sous forme de jeux quantitatifs, qui tiennent compte à la fois de l'énergie des capteurs et d'une valeur de gain incrémentée à chaque transmission réussie ; les graphes correspondants, les conditions de victoire sont fournis, et un exemple est développé sur le premier tour de jeu. La résolution générale des jeux produits est indécidable, mais elle est fournie pour un sous-ensemble d'objectifs constitués de conjonctions de littéraux. Cette modélisation peut être vue comme une ouverture sur

des travaux futurs, qui permettraient d'établir une méthode de détection plus efficace des capteurs compromis en évaluant leurs performances selon un ensemble de stratégies logiques.

Chapitre 8 : Conclusion Une synthèse des travaux abordés est réalisée avant de clôturer l'ouvrage : les différents mécanismes de sélection des *cNodes*, leurs principales caractéristiques, leurs cas d'utilisation sont passés en revue. Il est fait de même pour les différents outils de modélisation proposés. Les réseaux de PETRI utilisés, la logique stochastique, les jeux quantitatifs ont permis de créer des représentations formelles des systèmes étudiés, qui ne demandent qu'à être adaptées à d'autres cas d'utilisation. Mais d'autres pistes concernant de futurs angles d'approche sont présentées à la fin de cette conclusion, et offrent des perspectives sur les potentielles évolutions des travaux présentés dans cet ouvrage.

2

Architecture et enjeux des réseaux de capteurs sans fil

Puisque les travaux présentés dans cette thèse portent sur les réseaux de capteurs sans fil, il semble indispensable d'introduire tout d'abord les caractéristiques de ces capteurs et des réseaux qu'ils peuvent former. Le présent chapitre décrit donc le fonctionnement basique des capteurs, des réseaux, et fournit des exemples d'applications. Dans un second temps, ce sont certains des principaux problèmes soulevés par le déploiement des capteurs qui sont brièvement introduits, notamment la nécessité d'adapter les protocoles employés aux capacités des capteurs, les besoins éventuels de partition du réseau pour en obtenir une gestion plus efficace, ou encore des problèmes de mobilité, de sureté, de sécurité, de résilience.

1 Réseaux de capteurs sans fil

1.1 De quoi s'agit-il ?

Les réseaux de capteurs sans fil, ou *WSN* (pour *Wireless Sensor Networks* en anglais), sont des réseaux constitués de petits appareils, les capteurs, ainsi que d'une station de base. Les capteurs échangent par communications hertziennes, en utilisant des protocoles tels ceux définis dans la pile IEEE 802.11. Le routage des paquets dans le réseau peut faire appel à l'un des nombreux protocoles développés à cet effet (par exemple : AODV, OLSR), qu'il repose sur un algorithme centralisé (dirigé par une seule

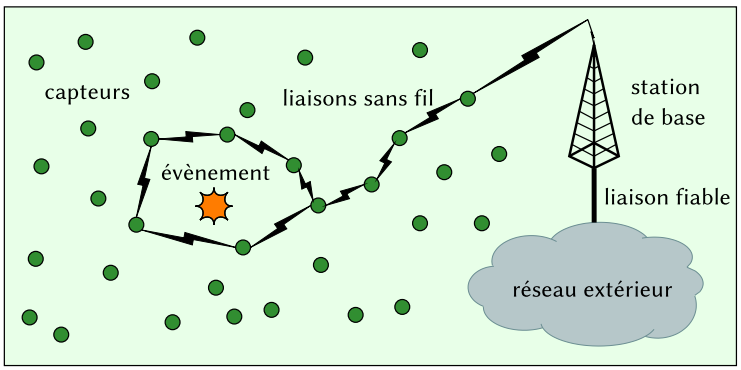

FIG. 2.1 : Schéma simple d'un réseau de capteurs sans fil

entité) ou distribué (exécuté par chaque entité du réseau). Les capteurs collectent des informations sur leur environnement et les font remonter à la station de base. Cette station de base, ou BS (pour *Base Station*), ou encore parfois *puits* (*sink* en anglais), est chargée de récolter et traiter les données provenant des capteurs. Une fois les capteurs déployés, l'administrateur n'interagit plus avec le réseau que par le biais de la station de base. Un schéma basique de réseau de capteurs sans fil est présenté en figure 2.1

Il est rare que tous les capteurs d'un WSN soient directement connectés les uns aux autres. À la topologie d'un réseau donné est donc très souvent associé le graphe de connectivité du réseau. Pour cette raison, il est souvent fait référence aux capteurs sous le terme de *nœuds* (ou de *nodes* en anglais).

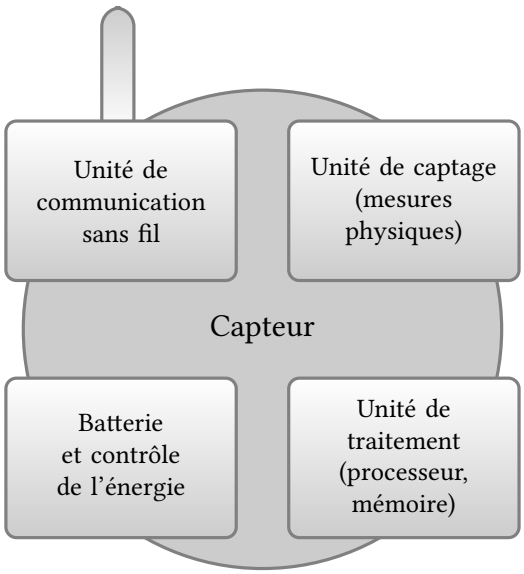

FIG. 2.2 : Schéma représentant les principaux modules d'un capteur standard

Un capteur seul peut être vu comme l'assemblage de quatre modules principaux, représentés en figure 2.2. Il s'agit :

- d'une unité de captage, chargée de réaliser des mesures physiques (analogiques)

sur l'environnement, puis de les convertir en un signal numérique ;

- d'un module de traitement, composé principalement d'un processeur et de mémoire (mémoire vive et mémoire non volatile), qui assure le fonctionnement du système d'exploitation, gère les interactions entre les différents modules, et surtout traite les données récoltées ;
- d'une unité de transmission/réception utilisée pour les communications hertziennes ;
- d'une batterie, accompagnée d'une unité de contrôle de l'énergie elle-même chargée d'alimenter de façon efficace les autres modules du capteur.

1.2 Applications

Le champ d'application des WSN est très vaste, et n'a de cesse de s'élargir au fil du temps et des avancées technologiques. Toutes ne font pas l'objet de publications scientifiques, mais certaines sont régulièrement évoquées en tant qu'exemples, ou bien font l'actualité dans le domaine des nouvelles technologies voire le domaine commercial.

Applications environnementales La gestion de l'environnement par l'Homme a de plus en plus recours à des mesures distribuées reposant sur l'usage de capteurs. Les prévisions météorologiques (reposant sur des mesures d'hygrométrie, de pression de l'air, *et cætera*) ont été l'un des premiers champs d'application des capteurs. Les mesures de qualité de l'air et de taux de pollution, en ville comme en campagne, se généralisent peu à peu. Les réseaux de capteurs permettent même de pousser la mesure de ce taux vers de nouveaux milieux, comme les glaciers [88] ou bien les océans.

L'agriculture est également susceptible d'avoir recours aux capteurs : des essais sont menés sur la réalisation de mesures faites par des microcapteurs semés en même temps que les cultures, permettant de surveiller au mieux les conditions de développement de ces dernières. Placés judicieusement dans l'habitat naturel de certaines espèces, les capteurs peuvent être utilisés pour suivre et analyser le comportement de la faune d'un milieu [73].

En forêt, les réseaux de capteurs peuvent permettre de détecter les départs de feu et de lutter plus efficacement contre les incendies. Et en zones à risques, ils peuvent être utilisés pour mesurer l'activité sismique ou volcanique du sol, et permettre une meilleure anticipation des phénomènes naturels. L'industrie a également recours à ces réseaux pour détecter d'éventuelles fuites de produits toxiques (pétrole, gaz, éléments radioactifs).

Il est à noter que certaines de ces applications touchent à des domaines critiques (en matière de sécurité des installations, voire de vies humaines) : il est indispensable d'assurer le bon fonctionnement du réseau déployé dans un tel contexte.

Surveillance et détection Les réseaux de capteurs sans fil sont aussi employés pour des mesures de sureté ou de sécurité, par exemple pour surveiller l'intégrité structurelle de certaines architectures (ferroviaires, aérospatiales, maritimes, ou plus simplement dans le bâtiment : gros œuvre, ouvrages d'art) et peuvent permettre une prévention efficace des pannes matérielles [106]. Mis à part la détection de pannes, les capteurs peuvent être utilisés dans des systèmes de vidéosurveillance en milieu urbain, ou sur des sites sensibles afin de prévenir les intrusions humaines, ou bien pour

détecter des incidents ou encore établir le suivi de certaines entités ; la gestion du trafic routier, ou même seulement des autobus, en sont des exemples. Le CEREMA (Centre d'études et d'expertise sur les risques, l'environnement, la mobilité et l'aménagement) a mis en ligne un site Internet particulièrement riche en exemples d'application des capteurs, et détaillant notamment les différentes technologies utilisées pour réaliser les mesures [24]. Pour exemple la seule détection de véhicules sur les voies de circulation peut reposer sur :

- la détection d'une modification du champ magnétique environnant ;
- la pression exercée par le véhicule sur la voie ;
- la variation de la pression dans un tube pneumatique ;
- les déformations de la structure d'un pont sous le poids des véhicules ;
- les variations d'intensité lumineuse ou sonore ;
- le renvoi d'ondes radar, ou lumineuses (laser), ou infrarouges, ou ultrasons ;
- l'analyse d'image vidéo visibles ou infrarouges, par reconnaissance de forme, ou bien par détection du mouvement des groupes de pixels…

Domaine militaire La surveillance et la détection sont aussi très utilisés dans le domaine militaire. Qu'il s'agisse de mener des opérations de renseignement ou bien de suivre l'évolution d'un combat sur le champ de bataille, pour analyser les cibles, relier entre eux fantassins et véhicules de tous types, pour détecter les agents radioactifs, chimiques ou biologiques répandus par un ennemi, toutes les informations sont bonnes à prendre [6]. Un maximum d'entre elles doivent être remontées au centre de commandement, afin de permettre une supervision optimale des forces en mouvement. Plus encore que les exemples précédents, l'usage des capteurs en contexte militaire implique de fortes contraintes quant à la sécurité du réseau déployé.

Médecine Certains usages des capteurs tiennent du domaine médical [120]. Il peut s'agir par exemple de faire communiquer un ou plusieurs implants avec un contrôleur externe. Il peut également être question d'employer des microcapteurs pour analyser avec précisions des variables corporelles (taux de glycémie, surveillance des organes vitaux) ou bien, sur une plus courte durée, pour traiter avec précision une pathologie locale (des cellules cancéreuses par exemple).

Grand public Le public a lui aussi de plus en plus accès à des applications reposant sur les capteurs. En domotique par exemple, la température des différentes pièces d'un lieu d'habitation peut être rendue accessible à tout instant, tandis que des fonctions comme l'activation de volets électriques, l'ouverture de portes, l'activation de sources lumineuses ou de dispositifs de chauffage peuvent être effectuées à distance par l'habitant (par exemple par le biais d'une application logicielle sur téléphone portable) ou bien de façon automatique en fonction des besoins identifiés à partir des données récoltées [6].

Un autre domaine d'application en voie de développement est ce que l'on appelle l'*Internet des objets* (*the Internet of things* en anglais), et qui consiste en quelque sorte à étendre Internet au monde réel, par le biais d'une interconnexion réseau entre les

objets de la vie courante : de plus en plus d'objets du quotidien tendent à devenir « connectés ». Ils se voient alors équipés de capteurs, d'un module de communication sans fil, et offrent de nouvelles possibilités en termes d'usages. Quelques exemples :

- les montres (interfacées avec les téléphones « intelligents » pour afficher des notifications diverses) ;
- les ampoules d'éclairage (gestion de la luminosité, de la couleur de l'éclairage, parfois dynamiques) ;
- les ceintures (accessoires vestimentaires : réajustement automatique du réglage, collecte de données sur le tour de taille pour le contrôle du poids) ;
- les réfrigérateurs (gestion des réserves alimentaires, création de listes d'achat) ;
- les véhicules autonomes, tels que les voitures sans conducteurs ou les drones (pilotage de l'engin, relevé de mesures tactiques pour les drones militaires).

Des puces RFID (de l'anglais *Radio Frequency Identification*, « identification sur fréquences radio ») se retrouvent par ailleurs de plus en plus utilisées, entre autres raisons pour faciliter de telles interactions dans le monde « connecté » [121]. Au fur et à mesure que ces objets intègrent la vie quotidienne des utilisateurs, ils auront probablement tendance à communiquer de plus en plus entre eux, entre objets, en ne centralisant les données que sur une seule interface présentée à l'utilisateur final.

Les exemples présentés ici ne représentent bien sûr que quelques applications des capteurs parmi les nombreuses qui existent [6], et dont la quantité ne cesse par ailleurs de croitre au fil du temps.

1.3 Contraintes en ressources

De par leur petite taille, et à cause de leur déploiement dans des zones souvent difficiles d'accès, les capteurs n'embarquent qu'une quantité limitée de matériel, qui ne peut pas toujours être remplacé. Les capteurs se retrouvent donc avec des capacités limitées [22], notamment en ce qui concerne :

- les **capacités de calcul** : les processeurs embarqués sont relativement peu puissants, principalement pour des raisons de cout. Les algorithmes exécutés par les capteurs doivent donc être de complexité relativement basse ;
- les **capacités de mémoire** : les capteurs disposent de mémoire vive (RAM, *Random Access Memory*, « mémoire à accès non séquentiel » en anglais) et d'un peu d'espace de stockage, mais ils ne sont pas du tout conçus pour sauvegarder de grandes bases de données. Les informations récoltées doivent être acheminées à la station de base, et non stockées sur le long terme par les capteurs eux-mêmes ;
- l'**énergie disponible** : les capteurs disposent d'une batterie qui leur fournit une quantité d'énergie finie, et (la plupart du temps) non rechargeable. Il est donc essentiel de conserver à l'esprit une gestion parcimonieuse de l'énergie pour tout programme implémenté sur les capteurs. Des calculs importants, ainsi que des émissions/réceptions d'ondes électromagnétiques nombreuses ou mal gérées, sont les principaux facteurs d'un épuisement prématuré de la batterie.

Il est à noter qu'au regard de ces contraintes qui affectent les capteurs, la station de base est considérée comme disposant de capacités « illimitées ».

1.4 Communications sans fil

Comme l'indique leur nom, les réseaux de capteurs sans fil n'utilisent aucun câble physique pour communiquer entre eux ou avec la station de base : toutes les transmissions sont effectuées par voie hertzienne. Chaque capteur est équipé d'un module radio utilisé alternativement pour émettre et pour recevoir. La plupart du temps ces modules sont capables de changer de fréquence de communication, ainsi que de moduler la puissance d'émission utilisée pour les transmissions.

Les protocoles de communication déployés sur cette architecture sont multiples (on parle d'*encapsulation* des données). Il faut pouvoir communiquer de pair à pair entre nœuds voisins, tout comme il faut être capable de faire parvenir un message à un nœud éloigné en faisant retransmettre les paquets par des nœuds relais successifs, d'assurer certains services comme le maintien de session, ou de gérer des applications. Comme dans la plupart des réseaux informatiques, l'empilement des protocoles reprend donc le modèle TCP/IP [121] (schéma concret lui-même issu du modèle théorique OSI, de l'anglais *Open Systems Interconnection*), présenté en figure 2.3.

		Exemples :
5	Application	HTTP, FTP, SSH
4	Transport	TCP, UDP
3	Réseau	IP
2	Liaison	IEEE 802.11 (CSMA/CA)
1	Physique	ondes électromagnétiques

FIG. 2.3 : Modèle TCP/IP

Certains standards normalisés définissent l'usage de protocoles spécifiques sur les trois premières couches. Les principaux standards qui sont employés dans les réseaux de capteurs sont les piles IEEE 802.11 (correspondant à la marque Wi-Fi) et IEEE 802.15.4 (sur laquelle sont basés la marque ZigBee et le standard IETF 6LoWPAN), et dans une moindre mesure la pile IEEE 802.15.1 (correspondant à la marque Bluetooth).

La couche physique Cette couche concerne l'émission et la réception en soi des ondes électromagnétiques, et l'encodage utilisé pour leur faire porter des valeurs numériques (par opposition à un signal analogique) [121]. Les réseaux sans fil ont la particularité de ne pas pouvoir restreindre l'envoi d'un paquet vers son seul destinataire : un paquet émis par transmission électromagnétique est reçu par tout nœud voisin (c'est-à-dire à distance suffisamment faible pour être couvert par la portée d'émission de l'émetteur) qui écoute le canal (donc les plages de fréquences concernées) au moment de la transmission. Il est parfois possible de se servir d'une antenne directionnelle pour réduire sensiblement les directions dans laquelle est réalisée la diffusion et pour augmenter le ratio de la puissance du signal émis sur l'énergie consommée par l'envoi ; mais cet équipement couteux n'est que très rarement utilisé dans les réseaux de capteurs.

Une propriété fréquemment employée en revanche est la capacité à moduler la puissance du signal en fonction des besoins. Plus le signal est fort, plus il portera loin (et permettra d'atteindre des nœuds distants). Naturellement, l'émission consommera également plus d'énergie.

La couche de liaison de données La deuxième couche du modèle fournit les moyens fonctionnels et procéduraux pour le transfert des données entre deux entités du réseau [121]. Elle permet aussi, le plus souvent, de détecter et éventuellement corriger certaines erreurs survenues sur la couche physique (en cas de perturbation ou dégradation du signal électromagnétique). Elle se décompose en deux sous-couches : la couche de contrôle de la liaison logique (LLC, pour *Logical Link Control* en anglais, « contrôle de la liaison logique ») et la couche du contrôle d'accès au support (MAC, pour *Media Access Control*, « contrôle d'accès au support »). LLC est la sous-couche haute, utilisée pour fiabiliser la sous-couche MAC ; elle intervient peu dans les réseaux de capteurs. Le protocole de couche MAC définit la manière dont les différents agents du réseau accèdent au médium de transmission de façon à limiter les collisions, et à garantir un accès le plus souvent équivalent au médium pour tous les nœuds. Les

Tab. 2.1 : Méthodes d'accès au médium de transmission

Nom anglais	Traduction	Description
Commutation de circuits et création de canaux		
Frequency Division Multiple Access (FDMA)	Accès multiple par répartition en fréquence	Plusieurs canaux basés sur des fréquences différentes
Code division multiple access (CDMA)	Accès multiple par répartition en code	Étalement du spectre de fréquence utilisé en conjonction avec des techniques comme les sauts de fréquence ou la génération de bruit pseudo-aléatoires (avec la même séquence pseudo-aléatoire appliquée au signal côté émetteur comme côté destinataire)
Time division multiple access (TDMA)	Accès multiple à répartition dans le temps	Un seul canal dont l'accès est réparti par créneaux dans le temps
Space division multiple access (SDMA)	Accès multiple à répartition dans l'espace	Plusieurs canaux spatiaux obtenus à l'aide d'antennes directionnelles. À noter : les antennes directionnelles augmentent sensiblement le cout de production des capteurs.
Modes d'accès par paquet		
Contention based random multiple access methods	Accès par contention	Contention par le nœud du paquet à envoyer jusqu'à ce que le protocole le lui autorise. Dans cette catégorie se trouve notamment le protocole CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*, accès multiple par écoute du canal avec esquive de collision), très utilisé dans les réseaux sans fil (IEEE 802.11 entre autres).
Resource reservation (scheduled) packet-mode protocols	Réservation des ressources	Réservation par un nœud des ressources (par exemple : créneau temporel) nécessaires à la transmission)
D'autres modes d'accès par paquet (*token passing*, *polling*) existent mais ne sont pas utilisés dans les réseaux de capteurs		

différents modes d'accès au médium existants sont résumés dans la table 2.1. Certains d'entre eux consistent à créer des canaux de transmission distincts, tandis que d'autres déterminent l'accès à une même bande de fréquence en instaurant des règles.

De ces modes d'accès sont dérivés de très nombreux protocoles de couche MAC, dont plusieurs ont été conçus spécifiquement pour les réseaux de capteurs [135]. Par exemple, en dehors des standards IEEE, le protocole S-MAC [136] fait alterner périodes « actives » et périodes de sommeil aux capteurs afin de préserver leur batterie. Les capteurs sont associés dans des groupes (qui ne correspondent pas tout à fait à des clusters) dont tous les membres sont en éveil de manière simultanée, afin de pouvoir communiquer à l'aide d'une version modifiée de CSMA/CA (par rapport à la version définie dans le standard IEEE 802.11).

La couche réseau Cette couche, pour les capteurs sans fil, repose le plus souvent sur le protocole IP (*Internet Protocol* en anglais), que ce soit en version 4 ou 6 (la version 6 est notamment utilisée avec la pile 6LoWPAN), pour l'adressage, et sur un protocole de routage qui détermine comment les paquets sont retransmis saut après saut dans le réseau [121].

Les réseaux de capteurs sans fil composent en quelque sorte une sous-famille des réseaux sans fil *ad hoc* (ou WANET, de l'anglais *Wireless Ad hoc Network*), et une grande partie des protocoles et algorithmes utilisés proviennent du monde des WANET. Des exemples classiques de ces algorithmes de routage incluent AODV (*Ad hoc On-Demand Distance Vector Routing*) [63] ou son évolution DSR [65] (*Dynamic Source Routing*), OLSR [64] (*Optimized Link State Routing Protocol*) ou ses évolutions DSDV (*Destination-Sequenced Distance Vector routing*) et B.A.T.M.A.N. (*Better Approach To Mobile Adhoc Networking*), *et cætera*. Ces protocoles se répartissent le plus souvent en deux grandes classes : les protocoles « proactifs » reposant sur une table de routage établie *a priori*, en amont des requêtes (AODV par exemple) ; et les protocoles de routage « à la demande » (dont OLSR), qui établissent les routes à la volée lorsqu'une requête de retransmission parvient au nœud relai.

La couche transport Cette couche gère les communications de bout en bout entre processus. Les protocoles les plus fréquemment utilisés, TCP (*Transmission Control Protocol*) et UDP (*User Datagram Protocol*) assurent l'ordonnancement des paquets et permettent d'éviter les pertes, les doublons et la corruption des paquets [121]. Le protocole TCP permet en plus à deux entités de communiquer en mode connecté en établissant une session (début, fin et validation des échanges). En raison des données de contrôle que ces protocoles impliquent, ils ne sont pas systématiquement implémentés dans les réseaux de capteurs (ils le sont parfois dans des versions allégées).

La couche application La dernière couche gère les échanges applicatifs entre les différents agents du réseau, et encapsule les données utiles au format désiré pour leur traitement final [121]. Il s'agit de la couche la plus haut niveau du modèle. Elle assure un service propre à l'application déployée, et les protocoles déployés à ce niveau dépendent totalement de l'objectif final du réseau, il n'y a donc pas de standard propre aux réseaux de capteurs.

1.5 Terminologie employée

Capteurs et nœuds Un réseau de capteurs sans fil est souvent représenté sous forme de graphe. En conséquence, il est souvent fait référence aux capteurs eux-mêmes sous le terme de « nœuds ». Nous parlerons plutôt de capteurs lorsque nous évoquerons les « réseaux de capteurs » eux-mêmes, ainsi que les mesures physiques qu'ils réalisent sur leur environnement, et plutôt de nœuds lorsque nous sommes amenés à travailler sur des graphes. Mais la plupart du temps, ces deux termes peuvent être employés dans cette thèse de manière complètement interchangeable. Leur alternance n'a le plus souvent que pour but de limiter les répétitions au sein d'une phrase.

Nœuds normaux Les « nœuds normaux » ont deux significations possibles selon le contexte.

1. Lorsqu'il est question de partition, de « clusterisation » du réseau, un nœud *normal* est un nœud qui n'a pas été sélectionné pour assurer la fonction de « chef » du cluster, de cluster head. De même lorsque des nœuds de contrôle (*cNodes* ou *vNodes*) ont été sélectionnés, les nœuds *normaux* sont les nœuds qui n'ont pas été sélectionnés (ni comme nœud de contrôle, ni comme cluster head) : ils poursuivent donc simplement leur rôle de collecte et de transmission des données mesurées.

2. Lorsqu'il est question de sécurité et d'attaques menées depuis l'intérieur du réseau, un nœud *normal* désigne un nœud non compromis. Les termes « nœud légitime », « nœud sain » ont le même sens dans cette thèse. Ils sont à opposer au nœud « attaquant », « compromis », « corrompu » ou occasionnellement « cupide ».

Attaquant Le terme d'« attaquant » fait par principe référence à l'entité consciente qui se trouve à l'origine d'une attaque, qu'il s'agisse d'une personne seule ou d'une organisation. Il arrive toutefois que par abus de langage, l'« attaquant » soit utilisé en tant que raccourci pour « le nœud attaquant », c'est-à-dire le nœud corrompu par l'*attaquant* réel, depuis lequel l'attaque est menée au sein du réseau.

Exploitant L'*exploitant* du réseau désigne l'entité (entité humaine ou organisation) qui exploite le réseaux de capteurs sans fil : l'entité qui interagit avec la station de base, de l'« autre côté » du réseau, pour récupérer et analyser les données collectées par les capteurs et remontées jusqu'à elle. Sauf précision contraire, on fait généralement l'amalgame avec l'*administrateur* du réseau, qui procède directement à sa mise en place et éventuellement à son maintien en fonction.

Des capteurs conscients Il va de soi qu'un capteur est un appareil dépourvu de vie et de conscience. Par abus de langage, des raccourcis sont parfois empruntés dans cette thèse et leur prêtent des intentions ainsi qu'une durée de vie. Ainsi il arrive que les capteurs « cherchent à », « essayent de », « veulent » réaliser des actions ; ou bien qu'ils « prétendent » être de meilleurs candidats au poste de cluster head. De même, ils sont susceptibles de « mourir » lorsque leur batterie est épuisée. Ces tournures, bien qu'abusives en termes d'exactitude du langage, permettent lorsqu'elles sont employées de simplifier les explications.

2 Problématiques liées aux réseaux de capteurs

De par leurs ressources faibles, leur dispersion dans un environnement parfois hostile, leurs communications sans fil, et les différentes missions qui leur sont confiées, les réseaux de capteurs sans fil introduisent une multitude de problématiques que l'on ne retrouve pas ou peu dans d'autres types de réseaux. Nous introduisons ici celles qui sont indispensables à la compréhension des travaux de cette thèse.

2.1 Gestion des ressources et performances

Les évolutions technologiques des dernières décennies n'ont de cesse de rendre les ordinateurs plus performants, plus compacts, plus rapides dans le traitement des données. Elles ont rendu possible, à force de miniaturisation et de réduction des couts, la création et le déploiement des réseaux de capteurs sans fil, mais ceux-ci restent soumis à des contraintes en ressources très fortes par rapport aux stations de travail « classiques » tel que les ordinateurs personnels (y compris portables).

2.1.1 ▶ Des algorithmes à adapter

Les algorithmes et protocoles déployés dans les réseaux de capteurs doivent être peu exigeants en matière de calculs. Il est même possible de réduire les calculs en amont, en sélectionnant judicieusement les échantillons de donner à mesurer puis à analyser, pour éviter des calculs superflus ; les algorithmes de traitement, d'agrégation de données, doivent également être pensés pour les capteurs [8].

Les mécanismes cryptographiques par exemple, souvent déployés pour des raisons de sécurité (et qui seront abordés plus en détail en chapitre 3), produisent par nature un grand nombre de calculs : les versions les plus complexes ne peuvent pas toujours être implémentées sur les réseaux de capteurs. Le choix des algorithmes à utiliser [78] en fonction de la mémoire ou de la vitesse du processeur disponibles et du niveau de sécurité requis, leur optimisation (et la création de nouveaux algorithmes) [45] aussi bien que leur impact sur la durée de vie des réseaux de capteurs [100] ont donc fait l'objet de plusieurs études.

Sur un autre niveau, il existe plusieurs études portant sur la mise en place de protocoles de couche de liaison de données, qu'il s'agisse de comparer la mise en œuvre d'un protocole comme celui de la pile IEEE 802.15.4 sur différentes architectures matérielles de capteurs [13], ou bien de comparer les multiples protocoles proposés spécifiquement pour les réseaux de capteurs sur cette couche [135].

L'évolution constante des technologies joue aussi son rôle dans ce domaine, et il y a fort à parier que, comme partout ailleurs en informatique, des composants de plus en plus performants seront disponibles au fil du temps pour les capteurs. Les récentes avancées sur le traitement du silicène pour la fabrication de transistors pourraient ouvrir d'ici quelques années des opportunités nouvelles [122].

2.1.2 ▶ Une gestion fine de l'énergie

L'usage d'algorithmes mal adaptés au réseaux de capteurs ne fait pas que diminuer les performances du réseau au niveau de la vitesse de traitement des données : si le processeur est sollicité davantage, il va drainer plus d'énergie. Hors il est essentiel de

se montrer économe sur l'usage de la batterie des capteurs, qui ne peut généralement pas être rechargée ni remplacée à moindre frais.

Par conséquent, divers travaux ont été menés pour réduire autant que possible la consommation énergétique. Cette thèse s'inscrit d'ailleurs dans cette lignée, pour le domaine spécifique de la lutte contre le déni de service. Mais à un niveau plus général déjà, les performances de chaque opération, qu'il s'agisse de la collecte des données, de leur traitement, de leur occupation en mémoire, et de toutes les étapes relatives à leur transmission, peuvent être optimisées dans le but d'économiser de l'énergie. Les protocoles utilisés, le cycle de travail des capteurs, et jusqu'à la topologie même du réseau peuvent avoir un impact sur la consommation énergétique des nœuds [8].

Par exemple, les réseaux de capteurs reprennent des protocoles de routage initialement développés pour les WANET, qui n'ont pas systématiquement les mêmes contraintes en énergie ; ont donc été proposés de nombreux algorithmes de routage destinés spécifiquement aux réseaux de capteurs. Ainsi ERAPL (*Energy-Efficient Routing Algorithm to Prolong Lifetime*, « algorithme de routage économe en énergie permettant de prolonger la durée de vie ») [140] repose sur l'usage d'algorithmes génétiques ainsi que d'une « séquence de collecte de données » permettant d'éviter les boucles et les doublons de transmission.

On trouve aussi des solutions qui proposent d'économiser l'énergie en établissant un classement des paquets à transmettre selon leur priorité, c'est-à-dire selon leur importance pour l'exploitant [117]. Ainsi les paquets prioritaires sont retransmis plus rapidement, tandis que ceux d'importance moindre peuvent attendre que le trafic soit dans des conditions telles que la transmission ne devrait consommer qu'un minimum d'énergie ; la phase d'attente peut également permettre l'arrivée en mémoire tampon d'autres paquets et de procéder à leur agrégation.

Comme pour les processeurs, les technologies intervenant dans la fabrication des batteries évoluent, et permettent d'augmenter peu à peu la longévité des nouveaux capteurs. Certaines solutions permettent parfois de recharger la batterie à moindre cout. Ce peut être le cas avec l'usage de cellules photovoltaïques intégrés aux capteurs. C'est aussi une solution qui commence à être utilisée pour recharger certains capteurs piézo-électriques placés sous la surface de voies de circulation automobile, qui peuvent générer une tension électrique au passage des véhicules grâce à la pression mécanique exercée par ces derniers sur les voies [24].

Il se pourrait même que dans un futur plus ou moins proche, les capteurs puissent se dispenser totalement de batterie : des propositions récentes, reposant sur un mécanisme appelé *ambient backscatter* (qui se traduit littéralement par « rétrodiffusion ambiante »), permettent de convertir, au niveau d'un capteur, un signal électromagnétique reçu (de la station de base par exemple) en un courant électrique suffisant pour alimenter l'appareil le temps de traiter le message et de retransmettre une réponse [81].

Mais cette technologie est encore limitée, et elle est loin d'être généralisée. En attendant, l'une des possibilités essentielles permettant de réduire la consommation en énergie est l'utilisation d'une architecture hiérarchiquement clusterisée dans le réseau.

2.2 Partition hiérarchique des nœuds du réseau

2.2.1 ▶ Partition du réseau

« Clusteriser » un ensemble d'éléments revient à le diviser en sous-ensembles appelés *clusters*. Dans le cas des réseaux de capteurs sans fil, cette partition permet d'obtenir un routage efficace des paquets, en adoptant la configuration suivante :

1. tous les nœuds réunis au sein d'un même cluster sont capables de communiquer directement entre eux « en un saut » (*one-hop transmission*) ;

2. lors de la partition, un unique nœud par cluster est désigné « chef » du cluster. Il est choisi, de façon déterministe ou bien aléatoire selon l'algorithme employé, parmi les nœuds « normaux » du cluster. Ce « chef » est appelé *cluster head* (CH), littéralement « tête de cluster » en anglais ;

3. lorsqu'un capteur quelconque d'un cluster souhaite faire parvenir des données à un nœud d'un autre cluster [1], ou bien à la station de base, il envoie ses paquets au cluster head de son cluster ;

4. le cluster head retransmet alors les paquets, soit directement à la cible s'il s'agit de la station de base et qu'il peut l'atteindre, soit « en plusieurs sauts » en passant par d'autres cluster heads (*multi-hop transmission*), jusqu'à atteindre le destinataire.

Le schéma d'un réseau clusterisé est présenté en figure 2.4.

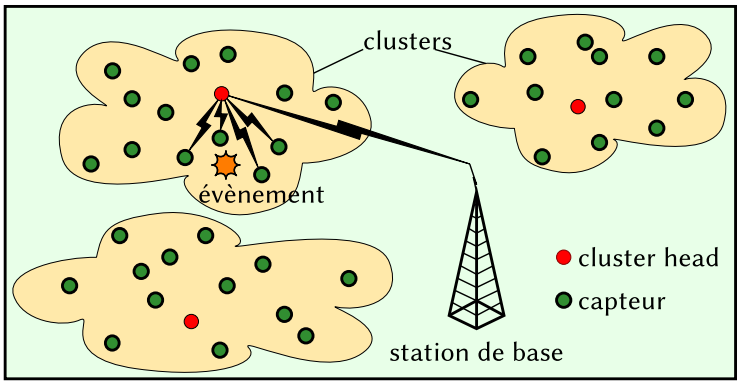

FIG. 2.4 : Schéma d'un réseau de capteurs clusterisé

L'appel à un algorithme de « clusterisation » a pour effet de limiter les émissions à « longue portée » (relativement aux communications intra-clusters) aux cluster heads seulement. Or les communications sur de plus grandes distances se traduisent par une plus grande consommation en énergie (puisqu'une plus grande puissance d'émission est nécessaire). Les capteurs normaux (non cluster heads) n'ont pas à atteindre directement des nœuds situés en dehors de leur cluster ; ils économisent d'autant en énergie.

[1] Pour une grande partie des applications, les communications entre nœuds de différents clusters, hors cluster heads, ne se produisent pas. La totalité du trafic « utile », acheminant des données, est généralement à destination de la station de base.

De plus, les cluster heads sont idéalement placés pour réaliser des opérations d'agrégation voire de compression sur les paquets qu'ils reçoivent, afin de limiter encore le volume des retransmissions couteuses en énergie.

À part les économies substantielles en énergie, la clusterisation d'un réseau présente plusieurs autres avantages :

- elle permet de déployer une gestion « centralisée » d'un cluster, puisque le cluster head est à même d'appliquer un algorithme tenant compte de tous les capteurs de son groupe. Pour autant, la topologie décentralisée de l'ensemble du réseau n'est pas sacrifiée, car les clusters sont indépendants de la station de base, qui n'intervient pas dans leur fonctionnement interne ;

- elle permet de gagner en extensibilité, puisqu'il est aisé de rajouter des clusters au réseau, voire peu contraignant de rajouter des nœuds dans un cluster donné. L'évolution du réseau est ainsi plus simple à assurer que s'il fallait modifier un algorithme distribué pour prendre en compte l'intégration de nouveaux capteurs.

2.2.2 ▶ Clusterisation hiérarchique

Une fois le réseau de capteurs divisé en clusters, rien n'empêche de considérer les clusters un à un et de leur appliquer à nouveau un algorithme de clusterisation, de façon à établir des sous-ensembles dans chaque cluster. Et ainsi de suite, de façon récursive, jusqu'à atteindre le degré de hiérarchie désiré. L'intérêt de cette méthode est de créer une partition hiérarchique dans le réseau, permettant un meilleur contrôle des sous-ensembles de capteurs. Par ailleurs, les clusters situés tout en bas dans la hiérarchie constituée seront de petite taille. Les communications intra-clusters seront donc peu consommatrices en énergie.

Pour pouvoir distinguer plus facilement le niveau de hiérarchie auquel nous nous plaçons, nous désignerons par la suite sous le terme k-*cluster* ($0 \leq k \leq$ nombre de capteurs) un sous-ensemble obtenu après k applications de l'algorithme de clusterisation. En suivant cette convention, l'unique 0-cluster est alors le réseau tout entier. Lorsque nous parlons simplement de *clusters*, il faudra comprendre 1-clusters ; autrement dit, des clusters issus d'une partition simple, sans degré supplémentaire de hiérarchie.

De même, on désignera par k-cluster head (ou bien par k-CH) les cluster heads de chacun des k-clusters du réseau. Le rôle de 0-CH pourra alors être attribué à la station de base. Chaque k-CH reçoit des données (provenant soit de nœuds normaux si k est le dernier degré de la hiérarchie constituée, soit de $(k+1)$-CH), les agrège et les transmet au $(k-1)$-CH auquel il est rattaché.

2.2.3 ▶ Un exemple : fonctionnement de l'algorithme LEACH

L'un des algorithmes de clusterisation les plus simples et les plus couramment employés dans les réseaux de capteurs sans fil est l'algorithme LEACH (*Low Energy Adaptive Clustering Hierarchy*, c'est-à-dire « hiérarchie de clusterisation adaptative à faible énergie » en anglais) [57]. Il s'agit d'un algorithme dynamique (il effectue de nouvelles clusterisations du réseau régulièrement dans le temps) qui, par la formation de clusters, met en place une solution de routage simple mais efficace des paquets dans le réseau.

Voici le fonctionnement détaillé de cet algorithme. Soit P le pourcentage moyen de CHs désirés parmi le total des nœuds dans le réseau à un instant quelconque t. LEACH est découpé (dans la durée) en cycles, chacun constitué de $\frac{1}{P}$ rondes. Chaque ronde r est organisée de la façon suivante :

1. Chaque nœud du réseau à partitionner calcule une valeur de seuil $S(i)$:

$$S(i) = \begin{cases} \dfrac{P}{1 - P \cdot \left(r \bmod \frac{1}{P}\right)} & \text{si } i \text{ n'a pas encore été CH} \\ 0 & \text{si } i \text{ a déjà été CH} \end{cases}$$

   Chaque nœud choisit un nombre pseudo-aléatoire $0 \leq x_i \leq 1$. Si $x_i \leq S(i)$, alors i s'auto-désigne comme CH pour la ronde en cours. Il est à noter que le calcul de $S(i)$ est réalisé de telle façon que chaque nœud devienne cluster head une fois et une fois seulement au cours de chaque cycle de $\frac{1}{P}$ rondes : le taux de probabilité d'auto-désignation $S(i)$ est égal à 1 lorsque la fin du cycle est atteinte (autrement dit, lorsque $r = \frac{1}{P} - 1$).

2. Les CHs auto-désignés informent leurs nœuds voisins de leur changement de statut à l'aide de messages en diffusion générale (*broadcast*). Tous ces messages sont envoyés en utilisant la même puissance de transmission (valeur fixe et prédéterminée lors de l'implémentation). Pour limiter les collisions, il est fait usage sur la couche MAC de la méthode *Carrier Sense Multiple Access* (CSMA).

3. Les autres nœuds, qui ne se sont pas désignés en tant que cluster heads pour la ronde en cours, choisissent de se joindre au cluster du CH dont ils perçoivent le signal avec l'intensité la plus élevée, c'est-à-dire le CH le plus proche quant à la puissance du signal électromagnétique reçu. Chaque nœud prévient le cluster head qu'il décide de rejoindre en lui envoyant un message. La méthode CSMA est là encore appliquée.

4. Au vu des réponses reçues, chaque cluster head calcule un « ordre de transmission » pour les nœuds qui l'ont rejoint. Il annonce alors à chacun de ces nœuds l'instant auquel le nœud doit lui transmettre ses données. Dans chaque cluster, les nœuds s'adresseront donc à leur cluster head à tour de rôle, selon l'ordre déterminé par le CH, ce qui revient à utiliser la méthode appelée *Time Division Multiple Access* (TDMA).

5. La phase de collecte des données peut débuter. Les cluster heads restent en écoute et reçoivent les données des autres capteurs de leur cluster. Les capteurs « normaux » effectuent leur mission (ils réalisent des mesures sur leur environnement), et envoient leurs résultats au cluster head lorsque c'est à leur tour de le faire. Quand ce n'est pas à leur tour de communiquer, ces nœuds mettent leur équipement radio en veille afin d'économiser leur énergie. Les collisions entre les transmissions des nœuds de différents clusters sont évitées grâce à la méthode appelée *Code Division Multiple Access* (CDMA).

6. Au fur et à mesure qu'ils reçoivent les données, les cluster heads agrègent, et éventuellement compressent ces dernières. Ils les envoient ensuite à la station de base, soit au cours d'une unique transmission directe, soit en faisant relayer les paquets par d'autres cluster heads.

7. Les étapes 5 et 6 sont répétées jusqu'à la fin de la ronde.

Quelques remarques sont à énoncer. Tout d'abord : pour un cluster donné, il est alors possible de réitérer l'application de l'algorithme LEACH, afin de créer une nouvelle partition au sein même d'un cluster. Et ainsi de suite par récursivité, jusqu'à obtenir le degré de hiérarchie désiré dans le réseau. Nous appellerons k-LEACH cet algorithme appliqué de façon à créer k degrés hiérarchiques.

Second point : l'un des aspects importants de LEACH est que lors de la première étape, chaque nœud choisi d'être, ou non, un cluster head pour la ronde en cours. Ce choix est basé uniquement sur la valeur de seuil calculée, et sur le nombre pseudo-aléatoire généré ; à aucun moment un nœud ne fait intervenir dans sa décision le comportement de ses voisins. En conséquence, le pourcentage P de cluster heads désirés dans le réseau n'est qu'une valeur moyenne sur l'ensemble des rondes de chaque cycle. Par ailleurs, la répartition géographique (au regard de la puissance de transmission nécessaire) idéale des cluster heads n'est en rien assurée. Au contraire, il est même probable d'obtenir, pour certaines rondes, une concentration importante de cluster heads dans une zone restreinte du réseau, tandis que d'autres régions seront mal couvertes. Il n'y a pas grand-chose à faire dans ce cas, sinon espérer que la prochaine ronde produira une distribution des CHs plus favorable. Si toutefois un nœud ne parvient à capter les messages d'aucun CH, il se déclare généralement lui-même cluster head.

2.2.4 ▶ De nombreux algorithmes de clusterisation

De nombreux algorithmes de clusterisation de données existent. Plusieurs d'entre eux sont même spécifiquement adaptés aux réseaux de capteurs sans fil. C'est le cas de LEACH qui prend place, comme évoqué plus haut, parmi les plus fréquemment utilisés, à tel point qu'il en existe de nombreuses variantes. Par exemple, il est possible d'étendre l'algorithme pour prendre en compte l'énergie restante dont dispose chaque nœud lors de l'élection des CH. Cette énergie résiduelle intervient alors en temps que paramètre supplémentaire lors du calcul de la valeur de seuil $S(i)$ [55]. D'autres travaux sont basés sur LEACH, que ce soit pour améliorer ses performances [103, 27] ou bien sa sécurité [95].

Mais on trouve également de nombreux autres protocoles [1, 35]. Certains d'entre eux prennent en compte deux à trois paramètres, comme l'énergie résiduelle des nœuds, leur distance aux cluster heads potentiels et/ou le nombre de voisins de ces derniers : c'est le cas du protocole HEED [137], assez souvent utilisé, ou du protocole MPC [74], plus récent et moins répandu, se basant sur les k-moyennes. Un quatrième élément, la confiance portée au nœud, est même utilisé parfois pour réaliser une sélection plus sure des CH [75].

Certains protocoles ont des buts plus précis, comme FFUCA [41, 40] qui repose sur l'exploitation de propriétés ultramétriques dans le réseau afin de créer une répartition « idéale » des nœuds dans les clusters en fonction de leur distance aux CH ; ou bien comme VSR [125], créé à l'intention des MANET, qui à l'aide d'une structure virtuelle détermine un routage proactif pour l'intérieur des clusters (où les nœuds émettent au cluster head en plusieurs sauts) et à la demande sur l'épine dorsale du réseau qui relie entre eux les CH [2].

[2] Pour conserver la lisibilité du paragraphe, le développement des sigles HEED, FFUCA, MPC et VSR n'est pas donné ici. Il est néanmoins disponible en fin d'ouvrage.

2.3 Déploiement autonome, mobilité

Les capteurs doivent être à même d'établir seuls une organisation autonome du réseau : ils comptent sur les protocoles inscrits dans leur mémoire, mais non sur une intervention externe (de l'administrateur par exemple). L'organisation présente un défi en soi [125], auquel les nombreux protocoles de découvertes des voisins et les algorithmes de routage proposés au fil des ans apportent autant de réponses [35]. Par ailleurs, ces protocoles sont tenus de gérer le départ ou l'arrivée de nouveaux nœuds dans le réseau. C'est ce que l'on appelle l'extensibilité du réseau (ou bien son évolutivité, ou encore sa scalabilité ; *scalability* est le terme anglais dénotant ce concept).

Qui plus est, il n'y a pas que le routage, mais l'intégralité de la chaine des opérations à réaliser qui doit être mise en place de façon autonome : la collecte de données en fait ainsi partie [140]. Un autre paramètre à gérer parfois est la mobilité des nœuds. Cette mobilité est une capacité propre aux MANET (*Mobile Ad hoc Networks*, « réseaux ad hoc mobiles »). Comme leur nom l'indique, il s'agit de WANET dont les nœuds sont mobiles. Lorsque ces nœuds sont (portés par) des véhicules, on parle même de VANET (*Vehicular Ad hoc Networks*, « réseaux ad hoc véhiculaires »). Ces réseaux ont leurs propres problématiques [33], qui sont souvent assez proches de celles des réseaux de capteurs sans fil. Certaines solutions développées pour les MANET, des mécanismes de sécurité par exemple, peuvent tout à fait être réutilisés dans les réseaux de capteurs [22].

Mais revenons à la mobilité en elle-même : elle n'est pas systématique dans les réseaux de capteurs sans fil, dont les nœuds sont la plupart du temps statiques ou quasiment statiques. Il arrive néanmoins que des réseaux soit constitués, en intégralité ou bien en partie, par des capteurs mobiles. Le fait de se retrouver dans un environnement où les distances et les vitesses évoluent continuellement rend accessibles de nouvelles applications, mais vient bien sûr compliquer le routage, la collecte des données, et même la gestion de la mobilité des nœuds (s'ils n'ont pas de pilotes humains par exemple, il faut éviter les collisions [139]). Dans le cas de réseaux de capteurs comportant des nœuds mobiles, le modèle du réseau se verra en général rattaché aux MANET et, bien que ne bénéficiant pas forcément des mêmes ressources, ce sont les mêmes algorithmes de routage qui vont être utilisés pour assurer l'acheminement des paquets de bout en bout. Nous reviendrons plus loin sur des solutions reposant sur un nombre restreint de nœuds mobiles se déplaçant au sein d'un réseau de capteurs fixes pour réaliser différentes opérations sur les appareils (collecte de données, diffusion d'information, réinitialisation de la configuration d'origine, *et cætera*) [52].

2.4 Sécurité, sureté et résilience

2.4.1 ▶ Sureté, résilience

Comme évoqué précédemment, une fois déployés, les réseaux de capteurs doivent s'organiser et fonctionner seuls, sans intervention humaine. Pour autant, ils ne sont pas à l'abri des erreurs ni des pannes. L'idéal est donc de pouvoir assurer un certain degré de sureté et de résilience au réseau.

La sureté d'un système est sa capacité à fonctionner sans erreurs, c'est-à-dire sans dysfonctionnement dus à une mauvaise conception ou à une mauvaise implémentation des architectures matérielle et logicielle. Elle est en principe apportée soit par des tests en amont du déploiement, soit par une modélisation formelle (ou semi-formelle) du système, qui permet d'en évaluer certaines propriétés et de déduire, en

fin de compte, si le système mis en place respecte ou non les spécifications établies.

Les tests et simulations sont récurrents dans la quasi-totalité des travaux de la littérature. Ils consistent à implémenter le dispositif étudié, physiquement ou virtuellement, pour étudier son évolution au cours d'une exécution « classique » ou bien soumise à des conditions d'expérimentation particulières. Certains travaux ont même pour objet des tests comparatifs sur différentes architectures matérielles de capteurs [100], ou encore recensent les différents systèmes de simulation permettant d'expérimenter sur les réseaux de capteurs [2].

Pour la vérification basée sur des méthodes formelles, des outils tels que le *model checking* peuvent par exemple être appliqués sur le modèle établi. Le *model checking* est une technique pouvant être utilisée sur des systèmes concurrents à états finis, et qui permet d'extraire automatiquement certaines propriétés sur les performances ou la sureté du modèle puis de s'assurer qu'elles sont respectées dans le réseau à tout instant. Si le système ne fonctionne pas comme prévu, l'outil permet parfois d'obtenir une trace qui vient faciliter la localisation de la source de l'erreur. Contrairement aux tests et aux simulations, les résultats obtenus à l'aide du *model checking* ne varient pas entre deux instances. Encore mieux : la spécification formelle du système permet de vérifier les propriétés sur tous les cas possibles d'exécution déterminés par les spécifications du modèle, tandis que des simulations ne permettent de remonter que certaines erreurs détectées (sans assurance de les avoir trouvées toutes). Le *model checking* est utilisé régulièrement dans les réseaux sans fils, que ce soit pour valider la conception de protocoles de routage comme pour la vérification de propriétés de sureté concernant les risques de collisions au sein d'un groupe de véhicules [139].

2.4.2 ▶ Résilience

La résilience d'un système est la capacité de ce système à outrepasser et éventuellement à corriger les erreurs susceptibles de survenir. Autrement dit, un système résilient est capable de poursuivre son fonctionnement avec toujours autant d'efficacité ou presque, même si des pannes matérielles, des erreurs logicielles ou d'autres types d'incidents viennent perturber le système. Assurer la résilience revient donc à anticiper, à prévoir des « plans de secours », des mécanismes de correction et de réparation automatiques.

Concrètement, dans un réseau de capteurs sans fil, elle met en jeu des principes comme la redondance des données stockées et envoyées, ou des routes de retransmission [119] ; elle s'établit sur l'usage de protocoles capables de se mettre à jour lorsque des nœuds disparaissent (épuisement supposé) ou apparaissent (nouveaux arrivants) dans le réseau, de mécanismes de correction d'erreurs… La résilience matérielle passe le plus souvent par la redondance des composants, mais n'est que rarement mise en place, pour des raisons de couts, dans les réseaux de capteurs. On préfèrera en général utiliser quelques capteurs en plus par rapport à la quantité strictement nécessaire pour assurer le service, sans avoir à augmenter le prix de *tous* les capteurs.

La sureté et la résilience des réseaux ne sont pas des problématiques de sécurité (car aucune attaque volontaire ne rentre en compte), mais il s'agit de problématiques assez proches de la disponibilité des réseaux (définie au chapitre 3, sous-section 1.1).

2.4.3 ▶ Sécurité

En raison de leurs faibles capacités, conjuguées aux domaines critiques dans lesquels ils interviennent parfois, les capteurs peuvent présenter des cibles de choix pour

un attaquant. Les algorithmes et mécanismes classiques utilisés en cryptographie afin d'assurer la confidentialité ou l'authentification des données sont souvent très exigeants en ressources (pour ce qui est des calculs notamment, et donc de la consommation énergétique). Il a fallu adapter ces mécanismes aux capteurs.

D'autres attaques peuvent être menées, non plus pour intercepter des données, mais dans le but de perturber le bon fonctionnement du réseau. Ce sont les attaques dites par *déni de service*. Un attaquant, par le biais d'un nœud compromis par exemple, peut ainsi chercher à détourner des flux de données au sein du réseau, à empêcher des paquets de parvenir à destination (en les supprimant), à maximiser le débit du nœud compromis au détriment de celui des autres nœuds, à saturer le canal de transmission pour empêcher les nœuds légitimes de l'utiliser, ou encore à pousser les autres capteurs à l'épuisement de leurs réserves en énergie, *et cætera*. Pour rappel, l'objectif des travaux présentés dans cet ouvrage consiste à proposer, modéliser et tester une solution permettant de détecter, puis de réagir aux attaques de ce type, tout en assurant une consommation minimale en ressources pour les capteurs du réseau.

Bien que le thème de la sécurité des réseaux de capteurs sans fil, notamment sur le sujet des attaques de type « déni de service » ainsi que des contre-mesures appropriées, ait tout à fait sa place dans ce chapitre, il s'agit d'un élément central pour les travaux de cet ouvrage, et il est développé avec beaucoup plus de précisions que les problématiques précédentes. Aussi a-t-il été choisi de lui attribuer un chapitre spécifique : les pages qui suivent abordent la question de la sécurité avec bien plus de détails que jusqu'à présent.

3

Sécurité dans les réseaux de capteurs sans fil

La sécurité dans les réseaux de capteurs sans fil est une qualité du réseau qui est souhaitée, au moins à un degré minimum, dans la totalité des applications. Certaines ont même des exigences plus fortes, et les protocoles utilisés doivent être à l'épreuve d'un grand nombre d'attaques. Mais comme mentionné précédemment, les faibles ressources des capteurs se prêtent assez mal à la mise en place des mécanismes traditionnellement déployés dans ce contexte, quand ils ne permettent pas tout simplement à l'attaquant d'exploiter de nouvelles failles. L'application de la sécurité aux réseaux de capteurs est donc un domaine spécifique qui est ici abordé en plusieurs temps. Ce sont d'abord les différentes composantes de la sécurité qui sont abordées : confidentialité, authentification, intégrité et autres propriétés sont introduites, avant de présenter les principales solutions correspondantes. Tout ce qui touche au déni de service fait l'objet d'un traitement à part et se trouve plus amplement développé en seconde partie du chapitre : y sont présentées les différentes classifications des attaques, les attaques à proprement parler, puis les solutions existantes pour s'en protéger, selon les trois axes suivants : prévention, détection et réaction.

1 Sécurité dans les réseaux de capteurs

1.1 Plusieurs problématiques

La sécurité des systèmes informatiques a pour particularité de constituer un domaine transversal à pratiquement tous les autres domaines de l'informatique, dans le sens où pour sécuriser la totalité d'un système, chaque couche, chaque technologie utilisée doit être sure. Ainsi, la sécurité des réseaux constitue en soi tout un domaine d'étude, qui se décompose à son tour en plusieurs problématiques. Les principales garanties que l'on cherche à apporter à un réseau sont les suivantes :

- la confidentialité des communications, c'est-à-dire qu'un attaquant ne doit pas être en mesure d'accéder au contenu utile des messages échangés ;
- l'intégrité des messages, autrement dit l'assurance qu'ils n'ont pas été modifiés par un attaquant durant leur transfert ;
- l'authentification des correspondants, propriété qui assure à un agent du réseau que son interlocuteur est bien celui qu'il prétend être ;
- la non-répudiation, qui empêche un agent de nier l'envoi d'un message émis ;
- la protection contre le rejeu, qui vise à prévenir les attaques basées sur l'injection de paquets capturés au préalable par l'attaquant dans le but de générer du trafic supplémentaire ;
- la sécurité physique des capteurs, nécessaire pour prévenir l'accès au contenu des communications ou au matériel cryptographique par exemple ;
- la disponibilité, c'est-à-dire la résistance aux attaques de déni de service. Cette problématique est centrale pour les travaux présentés ici, et sera développée plus largement que les autres dans la section 2.

À noter qu'il existe de nombreuses études, voire même des travaux de synthèse, qui se sont déjà concentrés sur la sécurité des réseaux de capteurs [33, 7].

1.2 Confidentialité

Les informations échangées sur un réseau sont dites confidentielles si un attaquant se retrouve dans l'impossibilité d'accéder au contenu utile de l'échange. On trouve en anglais les termes *data secrecy*, *privacy* ou encore *confidentiality* pour traiter de cette notion : la plupart des auteurs n'établissent pas de distinctions spécifiques entre ces expressions. Dans notre cas, il s'agit d'interdire l'accès aux données utiles des paquets à des agents extérieurs au réseau, ou bien même aux nœuds du réseau qui ne sont pas concernés par ces données et n'ont donc aucun besoin d'y accéder (afin de limiter les risques de fuites de données si l'un des nœuds devient compromis). Mais le terme *privacy*, qui se traduit par « vie privée », « intimité », présente aussi l'intérêt de s'appliquer également aux questions d'accès aux données : lorsque des capteurs sont utilisés sur des patients humains à des fins de surveillance médicale [120], il est

essentiel d'établir un contrôle sur l'accès aux données collectées : le patient et son médecin doivent y avoir accès, et il serait souhaitable que d'autres médecins urgentistes puissent y recourir au besoin ; tout autre accès doit impérativement être refusé. Cette notion ne sera pas abordée plus avant dans cette étude, nous conserverons un point de vue interne au réseau.

Les attaques portant sur la confidentialité des données sont multiples. L'attaquant peut adopter un comportement passif (simplement à l'écoute du trafic) ou bien actif (injection de paquets pour déclencher l'envoi de paquets en réponse, détournement de trafic). Pour de nombreux réseaux plus « classiques », l'utilisation d'un câble de transmission (Ethernet, fibre optique…) permet de poser une première barrière pour les écoutes passives. Mais dans les réseaux de capteurs, les données sont échangées par le biais d'ondes électromagnétiques que tout attaquant à portée peut recevoir. Il convient donc de trouver des solutions permettant de dissimuler le contenu des messages échangés, à défaut de prévenir l'accès aux messages eux-mêmes.

Les méthodes conventionnelles pour assurer la confidentialité des communications sont donc des solutions cryptographiques : des algorithmes de chiffrement symétrique (une même clé sert à chiffrer puis déchiffrer le message) ou bien asymétrique (une clé pour le chiffrement, une seconde pour déchiffrer) sont utilisés dans tous types de communications pour empêcher l'accès à l'information à quiconque ne possède pas la clé adéquate. AES (*Advanced Encryption Standard*) [93], basé sur l'algorithme Rijndael, en est l'un des exemples les plus célèbres et les plus utilisés aujourd'hui. Certaines études se sont penchées sur les vitesses d'exécution relatives de différents algorithmes utilisés pour le chiffrement et l'authentification des données [115].

Idéalement, pour des réseaux de capteurs, ces algorithmes doivent être implémentés de telle façon que soit respectées deux sous-problématiques liées au chiffrement [80] :

- la confidentialité persistante (*forward secrecy* en anglais, littéralement « confidentialité vers l'avant »), qui permet d'empêcher un nœud de déchiffrer les messages une fois qu'il a quitté le réseau ;
- la confidentialité « vers l'arrière » (*backward secrecy*), chargée d'assurer qu'un nouveau nœud entrant sur le réseau n'est pas en mesure de déchiffrer un message émis avant son arrivée.

Les mécanismes généralement impliqués tiennent bien sûr du domaine de la cryptographie, et doivent être intégrés aux protocoles de sécurité utilisés [36].

Les algorithmes de chiffrement nécessitent en général de nombreux calculs, voire beaucoup de mémoire tampon, pour être mis en œuvre, surtout lorsque l'appareil ne dispose pas de matériel dédié [100], ce qui les rend parfois peu adaptés aux réseaux de capteurs. Qui plus est, ils requièrent la plupart du temps la mise en place d'une architecture d'échange et de gestion du matériel cryptographique, notamment en ce qui concerne les échanges de clés : cette architecture peut considérablement alourdir la structure du réseau ainsi que les échanges en matière de volume de données de contrôle. Plusieurs solutions adaptées aux capacités limitées des capteurs ont donc été proposées [96]. Certaines d'entre elles demeurent basées sur des suites de protocoles dont les algorithmes de chiffrement sont choisis au mieux pour répondre aux capacités des capteurs [115, 72] ; ces solutions couvrent le plus souvent d'autres problématiques que la seule confidentialité des données, et certaines seront présentées dans la sous-section suivante. D'autres consistent à proposer de nouveaux algorithmes de chiffrement spécialement adaptés, tels que les blocs KLEIN [45]. Et d'autres encore

mettent en place des mécanismes propres à profiter de l'architecture particulière de ces réseaux, en voici certains exemples.

Dans le cas d'un attaquant « parasite » qui chercherait à exploiter les mesures relevées par le réseau de capteurs déployé par d'autres exploitants, des solutions comme GossiCrypt [83], basée sur un algorithme simple d'échanges de clés entre les nœuds et la station de base, ainsi que sur un renouvellement probabiliste du chiffrement : à chaque étape du message dans le réseau, le nœud relai chargé de renvoyer le message choisit aléatoirement de chiffrer à son tour, ou non, le contenu du paquet. Les clés de chiffrement des nœuds sont renouvelées régulièrement, mais pas toutes à la fois, de telle sorte qu'un nœud compromis par un attaquant ne devrait pouvoir déchiffrer le contenu que d'un nombre restreints de paquets.

Un usage intéressant de la cryptographie est le recours au chiffrement dit « homomorphique », qui permet d'effectuer certaines opérations sur des données chiffrées sans avoir besoin, justement, de les déchiffrer. Proposés pour les réseaux de capteurs [16], ces algorithmes permettent de procéder, dans une certaine mesure, à des opérations de compression et d'agrégation de données (par exemple) au cours du trajet d'un paquet, sans avoir à déchiffrer puis chiffrer à nouveau son contenu à chaque étape.

À part les solutions de chiffrement, d'autres méthodes consistent plus simplement à compliquer l'accès aux paquets plutôt qu'à chiffrer le contenu. Moins robustes, ces solutions sont aussi moins couteuses pour ce qui concerne les calculs et, *in fine*, la consommation énergétique. Certaines méthodes proposent ainsi de découper les paquets à envoyer en fragments, qui seront injectés à travers le réseau par des routes distinctes. Ces fragments seront combinés de telle sorte que seule la détention d'un certain nombre d'entre eux permette de reconstituer le message original. Cette combinaison des fragments peut faire appel à des mécanismes très simples, par exemple un « ou exclusif » logique comme avec la méthode SDMP (*Securing Data based on Multi-Path routing*) [17], ou bien à un mécanisme un peu plus élaboré de partage de secret entre plusieurs entités, comme l'a proposé par exemple Schamir [111]. Ces méthodes ont pour intérêt de gêner l'attaquant pour la reconstruction des messages dont il ne dispose que de quelques fragments. Elles sont inefficaces en revanche si l'attaquant contrôle toutes les routes du réseau, ou simplement s'il se trouve très proche géographiquement de l'émetteur ou du destinataire pour l'échange concerné. Plusieurs de ces méthodes peuvent être associées, pour ajuster au mieux le compromis entre la sécurité et l'économie des ressources. Nous avons ainsi proposé une solution pour assurer la confidentialité des échanges basée sur un système de priorités [19] :

- les paquets d'importance moindre peuvent être simplement découpés en fragment et envoyés par plusieurs routes distinctes selon la méthode SDMP ;
- les paquets dont le contenu est plus important peuvent se voir rajouter une couche de sécurité, comme avec l'usage de la méthode de secret partagé ;
- enfin, les messages d'importance critique, moins fréquents, utilisent des solutions plus lourdes, mais plus efficaces, de chiffrement cryptographique, qui seules sont à même d'apporter dans ce cas un degré convenable de sécurité.

Il est à noter qu'aucune des méthodes présentées ici, qu'elles reposent ou non sur l'usage de chiffrement, n'empêchent un attaquant de pouvoir récolter des métadonnées sur les échanges effectués, et d'en déduire par exemple qui communique avec qui, ou quel est le volume des données échangées. Ces métadonnées n'ont pas toujours une valeur importante dans les réseaux de capteurs, et peuvent être en partie

masquées, au besoin, par l'envoi de faux messages supplémentaires pour « brouiller les pistes », mais cela s'avèrerait très couteux pour le réseau.

1.3 Intégrité, authentification, non-répudiation, rejeu

1.3.1 ▸ Les concepts

Assurer l'intégrité d'un message revient à s'assurer que le contenu utile du message n'a pas été altéré ou amputé entre sa création par l'émetteur et la réception par le destinataire final. Des méthodes cryptographiques peuvent être employées pour créer un condensat ou (s'il est utilisé avec une clé symétrique) un code d'authentification (MAC en anglais, pour *Message Authentication Code*) des messages, permettant cette vérification.

L'authentification sert à prouver la validité de l'expéditeur des messages. Il n'est pas suffisant, pour assurer la sécurité d'un réseau, de s'assurer que le contenu des messages est chiffré : il faut également s'assurer que l'entité avec laquelle on communique est bien qui elle prétend être. Cette précaution permet notamment de se prémunir de l'attaque de *l'homme du milieu*. La signature des messages est le plus souvent réalisée à l'aide de mécanismes cryptographiques asymétriques (condensat signé à l'aide d'une clé privée). L'authentification, dans les réseaux de capteurs sans fil, regroupe en réalité à la fois l'authentification de deux parties l'une envers l'autre, et l'authentification d'un agent aux yeux de tous les autres membres du réseau, dans le cas d'une diffusion générale (*broadcast*) des messages. Selon le cas, les réseaux de capteurs peuvent également avoir besoin qu'un nœud s'identifie de façon spécifique auprès de la station de base, ou bien encore, pour certains réseaux clusterisés, qu'il puisse s'authentifier auprès des autres membres du cluster (et prouver son appartenance à ce dernier). Un système d'authentification efficace doit par ailleurs répondre à de nombreuses sous-problématiques [76] (non développées ici en détail).

La non-répudiation consiste à assurer qu'un agent du réseau ne peut plus nier avoir émis un paquet une fois celui-ci reçu par son destinataire. Concrètement, si un agent est le seul à posséder une clé privée utilisée pour la signature d'un paquet, il ne fait nul doute, une fois ce paquet reçu, que cet agent en question est bien à l'origine de la création du paquet, puisque nul autre agent du réseau n'est en mesure de le signer à sa place. L'usage de ces dispositifs de signature en font un aspect étroitement lié à l'authentification des messages. Indispensable dans les transactions en ligne, la non-répudiation n'est pas toujours essentielle dans les réseaux de capteurs, mais peut s'avérer utile pour identifier de manière certaine l'auteur de comportements malveillants dans le réseau.

Les attaques par rejeu consistent pour un attaquant à capturer des paquets, même chiffrés, en transit sur le réseau, puis à les injecter à nouveau dans le réseau. Le but visé peut être de générer davantage de trafic, dans le but d'augmenter le volume de données capturées. Il s'agit par exemple de la méthode utilisée pour accélérer considérablement l'obtention de la clé d'accès à un réseau sans fil sécurisé à l'aide de la suite WEP (*Wired Equivalent Privacy*) [21]. Un autre but recherché peut être de générer du trafic superflu dans le but de consommer inutilement les ressources du réseau, ce qui en fait potentiellement une attaque de déni de service. Pour se prévenir de ce genre d'attaques, des moyens cryptographiques peuvent ici encore être mis en œuvre pour associer un identifiant unique à chaque paquet, de sorte qu'un paquet rejoué soit rejeté par les agents du réseau.

Ces différentes propriétés sont distinctes, mais elles reposent presque toujours sur l'usage de solution cryptographiques pour être préservées dans un réseau. Elles sont souvent liées entre elles, ainsi qu'à la confidentialité des données, car il est fréquent qu'un dispositif mis en place pour assurer l'une de ces propriétés permette également d'en assurer d'autre. Par exemple, l'usage d'un condensat signé permet d'assurer à la fois l'intégrité d'un message, la validité de son auteur, et d'empêcher ce dernier de répudier le message dans le futur.

1.3.2 ▶ Les solutions

Une solution intéressante pour économiser des ressources consiste à n'introduire l'usage de solutions cryptographiques que lorsqu'elles deviennent nécessaires, et à s'en dispenser lorsque le réseau fonctionne correctement. Ce système a été proposé en association avec l'usage d'un « arbre d'agrégation sécurisée », structure du réseau utilisée pour le routage des paquets [133]. Lorsqu'un comportement malveillant est détecté dans le réseau, cet arbre permet de mettre en place rapidement et avec peu de données de contrôle un mécanisme d'authentification et de chiffrement des messages. Le danger de cette proposition repose sur la capacité du réseau à détecter les attaques : si aucun comportement suspect n'est constaté, aucune mesure de sécurité n'est mise en place. En cas d'écoute passive de la part de l'attaquant notamment, aucune mesure n'est appliquée pour protéger la confidentialité des données échangées.

Plusieurs propositions concernent la mise en place de solutions classiques, mais « allégées » [56], comme avec la mise en place d'une autorité de certification spécifique permettant l'authentification réciproque des nœuds deux à deux [50], ou bien basé sur des clés pré-distribuées [14]. Le protocole MSQPS (*Modified Secured Query Processing Scheme*) [44] a été proposé pour sécuriser le transfert des données dans un réseau clusterisé à travers un découpage des échanges sous forme de requêtes et de réponses authentifiées qui contiennent des marqueurs temporels, et ne peuvent être réinjectés dans le réseau, ce qui le rend efficace pour protéger les nœuds des attaques par rejeu.

De manière générale, les problèmes d'authentification, et dans une moindre mesure les autres problématiques soulevées dans cette sous-section, ont mené à la proposition de nombreuses architectures dédiées.

1.3.3 ▶ Architectures de sécurité

Dans cet ouvrage, une « architecture de sécurité » désigne un ensemble de protocoles et de mécanismes déployés dans le réseau en vue d'apporter certaines garanties en matière de sécurité. L'équivalent anglais serait *security framework*. Les solutions proposées sont multiples [51, 49, 109] ; nous allons présenter ici les plus connues.

Ainsi SPINS (*Security Protocols for Sensor Networks*) [99] est l'une des premières solutions proposées dans la littérature pour sécuriser les échanges dans les réseaux de capteurs, et sert de base pour de nombreuses solutions plus récentes. Il s'agit d'un système reposant sur deux blocs : SNEP (*Secure Network Encryption Protocol*) fournit la confidentialité, l'authentification entre deux parties, l'intégrité et la protection contre le rejeu, le tout en préservant un faible volume de données de contrôle en entête des paquets. Des codes d'authentification des messages associés à un compteur partagé entre les interlocuteurs, et utilisé pour le bloc de chiffrement (le compteur est incrémenté après chaque bloc), sont employés à cette fin. Le second bloc est appelé μTESLA (version « micro » du *Timed, Efficient, Streaming, Loss-tolerant Authentication Protocol*), il fournit des diffusions générales (*broadcast*) authentifiées à l'aide de

chaines de clés : ces chaines à sens uniques fournissent des clés associées à chaque intervalle de temps pour la signature des messages. Les paquets ainsi envoyés pendant l'intervalle en cours sont signés de manière symétrique à l'aide de la clé, tenue secrète, correspondant à l'intervalle temporel ; à la fin de cet intervalle, la clé est envoyée à tous les correspondants qui peuvent alors vérifier la légitimité des paquets enregistrés pendant la période. Le module SNEP a fait l'objet d'études comparatives sur les performances que lui confèrent différents algorithmes de chiffrement [110].

TinySec [71] se réfère à SNEP, mais lui reproche une spécification incomplète. Cette architecture propose l'authentification seule, ou bien accompagnée de chiffrement, des paquets. Elle écarte volontairement la protection contre le rejeu, mais se penche très sérieusement sur l'implémentation d'une solution classique basée sur un algorithme de chiffrement en conjonction avec un code d'authentification des messages (MAC). Les algorithmes possibles de chiffrement, leurs différents modes, leurs vecteurs d'initialisation, les codes d'authentification, les mécanismes d'échange de clés possibles sont tous analysés en détail, afin de déterminer les avantages et les inconvénients de chacun pour un réseau de capteurs.

MiniSec [82] est basé sur TinySec. Au mode CBC (*Cipher Block Chaining mode*, mode par enchainement des blocs), cette architecture préfère l'utilisation du mode de chiffrement OCB (*Offset CodeBook mode*, « dictionnaire avec décalage »), qui réalise à la fois le chiffrement et la signature des données et ne nécessite donc pas de code d'authentification distinct. Elle adopte une approche particulière du vecteur d'initialisation pour le chiffrement, qui lui permet de minimiser le volume des données d'entête. Enfin elle fait une distinction entre les diffusions nœud à nœud et les diffusions générales, pour lesquelles un filtre de Bloom est utilisé, avec pour objectif final de réduire sensiblement l'énergie consommée lors des envois.

TinyKey [30] est une autre architecture basée sur TinySec, mais plus récente. Elle reprend l'architecture de TinySec, mais rajoute un numéro de séquence aux paquets pour éviter les attaques par rejeu. Surtout, là où les auteurs de TinySec se contentent de passer en revue les mécanismes de gestion des clés qui peuvent être utilisés, les auteurs de TinyKey proposent une implémentation concrète de cette gestion, qui est assurée par deux modules centralisés sur la station de base : KMS (*Key Management Submodule*, « sous-module de gestion de clé » en anglais) et KDM (*Key Distribution Manager*, « gestionnaire de distribution des clés »).

D'autres architectures encore se concentrent sur la gestion des clés dans le réseau. Avec LEAP (*Localized Encryption and Authentication Protocol*, « protocole de chiffrement et d'authentification localisés ») [141], la gestion est décentralisée, et porte sur quatre types de clés :

- des clés individuelles permettant de communiquer de façon sécurisée avec la station de base ;
- une clé de groupe, utilisée par l'ensemble du réseau, pour protéger des diffusions générales d'un attaquant extérieur ;
- des clés de cluster, à utiliser, le cas échéant, au sein d'un cluster du réseau ;
- des clés pour les communications pair à pair.

Les mécanismes déployés ont pour but, là encore, de minimiser l'énergie consommée. Ils ont été améliorés dans une seconde version du protocole, baptisée LEAP+ [142].

L'architecture pDCS (*privacy-enhanced Data-Centric Sensor networks*, « réseaux de capteurs centrés sur les données, de confidentialité améliorée ») [112] divise le réseau

en cellules rectangulaires à l'aide d'arbres euclidiens de Steiner, et introduit des clés de cellules et de rangées. Des capteurs se trouvent dans les cellules, tandis que des « puits » de données mobiles se déplacent et récoltent les informations. Mais le chiffrement est fait de telle sorte qu'un puits illégitime ne soit pas en mesure de remonter au capteur qui a initialement détecté un évènement donné. Un filtre de Bloom est là encore utilisé pour minimiser le volume de données de contrôle produites. pDCS a été améliorée avec ERP-DCS (*Efficient Rekeying Protocol for DCS sensor networks*, « protocole efficace de régénération de clés pour les réseaux DCS ») [60], qui se concentre sur l'amélioration du mécanisme de gestion des clés qui intervient dans le cas où un agent a été compromis et identifié comme tel, en utilisant à cette fin un système d'exclusion appelé EBS (*Exclusion Basis System*).

ZigBee [143] est un protocole basé sur la pile IEEE 802.15.4, parfois utilisé dans les réseaux de capteurs, bien que son orientation soit plus large. Il est en quelque sorte actuellement candidat pour devenir un standard pour les objets connectés dans le contexte d'un « Internet des objets ». Utilisé avec les capteurs, il apporte chiffrement, authentification, protection contre le rejeu. La gestion des clés est centralisée (ZigBee introduit une notion de « centre de confiance »). La sécurité apportée vient toutefois au prix d'en-têtes et de calculs qui peuvent être plus importants qu'avec d'autres architectures.

La table 3.1 compare brièvement les fonctionnalités apportées par ces différentes modèles.

Tab. 3.1 : Brève comparaison d'architectures de sécurité pour réseaux de capteurs

Architecture	Mode de chiffr.	Algor. de chiffr.	Clés utilisées	Sécurité apportée
SPINS	CTR	RC5	Clé maitresse, chaine de clés symétriques	* : Chiffrement, authentification et intégrité des données, protection contre le rejeu
TinySec	CBC	Au choix	Au choix	(*) moins protection contre le rejeu
TinyKey	CBC-MAC	SkipJack	Clés pré-installées sur les nœuds	(*)
MiniSec	OCB ou CBC-MAC	SkipJack ou RC5	Clés pré-installées sur les nœuds	(*)
LEAP	RC5	RC5	Clé individuelle, clé globale, clé de cluster, clés pair à pair	(*)
LEAP+	RC5	RC5	Clé individuelle, clé globale, clé de cluster, clés pair à pair	(*)
pDCS	RC5	RC5	Clé maitresse, clés pair à pairs, clés de cellule, clés de rangée, clé globale	(*) plus dissimulation de la provenance des données
ERP-DCS	RC5	RC5	Clés pair à pair, clé de cellule, clé EBS	(*) plus dissimulation de la provenance des données
ZigBee	CBC-MAC	AES, modes CTR ou CCM	Clé maitresse, clés pair à pair, clé globale	(*)

Une première observation à réaliser concerne l'importance de la gestion des clés, qui occupe une place essentielle dans la plupart des systèmes proposés : il est en effet indispensable de disposer d'un moyen d'échanger, et au besoin de renouveler les clés nécessaires à ces architectures, sans compromettre leur sécurité mais sans pour autant consommer trop de ressources. De nombreux mécanismes ont été proposés et continuent de l'être [36, 14]. Nous ne les détaillerons pas ici ; mais il reste possible, pour en savoir davantage, de se tourner vers des travaux de synthèse qui catégorisent et décrivent les principales approches [56, 134].

Seconde remarque : dans la littérature, plusieurs des mécanismes introduits dans les architectures complexes présentées ci-dessus ont été repris pour des solutions ultérieures [109]. Par exemple, il existe une proposition concernant la combinaison de signatures numériques (cryptographie asymétrique), du bloc μTESLA (asymétrie temporelle) introduit avec SPINS mais sans reprendre SNEP, et d'un filtre de Bloom pour créer un mécanisme d'*Information Asymmetry*, qui assure des diffusions générales (*broadcast*) authentifiées [118]. Le filtre de Bloom est utilisé pour condenser plusieurs codes d'authentification de messages (MAC), qui sont ajoutés aux paquets pour l'authentification des transmissions par les nœuds. Plusieurs codes sont vérifiés par chaque nœud, qui possède son propre sous-ensemble de clés, de sorte que si certaines clés ont été compromises, il en reste d'autres à vérifier. Le bloc μTESLA en particulier est par ailleurs repris dans de nombreuses propositions, notamment avec le protocole SecLEACH qui l'introduit dans un réseau clusterisé de façon hiérarchique pour combler certains manques de l'algorithme LEACH en matière de sécurité [95].

À noter enfin que les architectures LEAP et TinyKey ont été analysées à l'aide d'un outil de *model checking* appelé AVISPA, spécifiquement conçu pour vérifier les protocoles de sécurité [127].

1.4 Sécurité physique

L'accès au réseau pour un attaquant peut passer par la compromission de l'un des nœuds. Dans le pire des cas, tout le système peut être modifié et détourné de son but d'origine, pour mener d'autres attaques par la suite, tandis que la mémoire, et notamment le matériel cryptographique embarqué, peuvent être entièrement analysés.

Les systèmes d'exploitation utilisés par les capteurs sont généralement conçus pour être légers et s'adapter aux contraintes des capteurs. Mais ils ne rajoutent pas de problématiques particulières en matière de sécurité système par rapport à d'autres machines, et le peu de fonctionnalités dont ils disposent joue plutôt en leur faveur pour ce qui est de la sécurité logicielle. Leur faiblesse réside donc dans leur déploiement en milieu ouvert et potentiellement hostile, ce qui peut permettre aux attaquants d'y accéder physiquement et de manière prolongée.

L'une des solutions proposées consiste donc à considérer un nœud comme ayant épuisé sa batterie s'il ne répond plus pendant une période prolongée ; si en revanche le capteur redevient actif par la suite, on suppose que son absence était due à son extraction du réseau par un attaquant, et le capteur est alors considéré comme compromis [58].

La compromission d'un nœud peut être extrêmement nuisible pour le réseau de capteurs. Non content d'accéder aux données reçues par ce nœud, ou aux clés de chiffrement et de signature qu'il peut contenir, l'attaquant peut en plus chercher à mener des attaques de déni de service visant à perturber le bon fonctionnement du réseau.

2 Déni de service

Une attaque dite par « déni de service » menée dans un réseau informatique est une attaque réalisée dans le but de nuire au fonctionnement normal de ce réseau. Il existe de très nombreuses façons de procéder, et on recense par conséquent une multitude d'attaques par déni de service existantes. L'état de l'art dans ce domaine (et à propos de la sécurité en général, par ailleurs) a ceci de particulier qu'il comporte deux points de vue : celui de l'attaquant et celui du « défenseur ». Il est indispensable de pouvoir définir le modèle d'une attaque pour pouvoir proposer des contre-mesures adéquates. Et de façon plus ou moins réciproque, les mécanismes de protection mis en place au fil du temps poussent les attaquants (ou les chercheurs) à développer de nouvelles attaques pour les contourner.

Les réseaux de capteurs se retrouvent malheureusement très exposés aux attaques de déni de service [102], du fait de :

- leurs ressources extrêmement limitées, et principalement en termes d'énergie ;
- leurs faibles capacités, qui peuvent introduire des délais (latence dans les communications ou délai de traitement)
- leur exposition aux attaques physiques ;
- la faible fiabilité du médium de transmission, au sujet de la confidentialité ou des collisions ;
- leur gestion réalisée à distance ;
- l'absence d'une gestion centralisée (et l'impossibilité de connaitre avec précision le statut des autres nœuds) ;

Dans cette section nous présentons les principales attaques répertoriées dans les réseaux de capteurs, puis nous abordons les mécanismes proposés en réponse dans la littérature. Auparavant, nous allons voir qu'il existe plusieurs façons de classer ces attaques.

2.1 Différentes classifications

2.1.1 ▸ Selon l'objectif recherché

Une première façon de classer les attaques est de considérer l'objectif de l'attaquant. Les buts principaux de ces attaques sont :

- l'accaparement de ressources pour les besoins propres de l'attaquant, au détriment des autres agents du réseau (par exemple, monopolisation du canal de transmission pour l'envoi des données de l'attaquant exclusivement). Il s'agit de comportements « cupides » (*greedy* en anglais) ;
- la réduction, voir l'annihilation de la capacité du réseau à assurer correctement les services pour lequel il a été déployé, afin de nuire aux exploitants de ce réseau. Cette nuisance peut s'exprimer par des pertes financières, par exemple lorsqu'une telle attaque est menée sur Internet contre un site de commerce en

ligne ; dans le cas des réseaux de capteurs sans fil, et notamment lorsqu'ils sont utilisés dans un cadre militaire, l'exploitant d'un réseau peut alors se voir priver de l'accès aux informations stratégiques que devaient récolter les capteurs. On parle le plus souvent d'attaques de type *jamming* en anglais, qui se traduit selon le cas par « brouillage », « encombrement », mais d'autres attaques reposant sur la privation de sommeil des capteurs, ou bien leur destruction physique, peuvent aussi être réalisées dans cet objectif ;

- plus rarement, l'induction en erreur de l'exploitant du réseau de capteurs. Pour ceci l'attaquant cherche à fausser les résultats collectés (changement d'environnement des capteurs) ou transmis (altération du contenu ou du volume de données transmises).

2.1.2 ▶ Selon la situation de l'attaquant

Les attaques peuvent également être différenciées selon la provenance de l'attaquant, à savoir s'il fait partie du réseau, et mène par exemple une attaque sur le protocole de routage employé, ou bien s'il agit depuis l'extérieur du réseau, depuis une machine qui n'est pas reconnue comme faisant partie du réseau de capteurs [120]. Ce deuxième cas peut être illustré par un attaquant qui chercherait à brouiller de manière globale toutes les fréquences radio utilisées par les capteurs pour communiquer.

Un attaquant extérieur n'a pas forcément connaissance de la façon dont fonctionne le réseau (architecture, protocoles employés, mesure de détection mises en place). Il peut mener une attaque sans ces informations (cas du brouillage des fréquences par exemple), ou bien justement chercher à s'introduire dans le réseau. Cette intrusion peut être réalisée par l'attaquant soit en faisant accepter l'un de ses propres appareils aux autres agents du réseau, soit en compromettant l'un des agents jusqu'alors légitime dans le réseau. Une fois l'accès interne obtenu, il devient généralement beaucoup plus facile d'accéder à des informations sur le fonctionnement du réseau. L'attaquant, en fonction des mesures mises en place, devient aussi susceptible d'être détecté et exclu du réseau. Les attaques menées depuis l'intérieur jouent souvent sur les paramètres des protocoles employés, et sont souvent plus « subtiles », plus délicates à détecter et identifier pour l'opérateur du réseau si aucune méthode de détection d'intrusion n'a été mise en place.

2.1.3 ▶ Selon les capacités de l'attaquant

La puissance dont dispose l'attaquant, que ce soit en matière de calcul, d'émissions électromagnétiques, ou bien d'alimentation en énergie, est un autre moyen de classifier les attaques [7]. Un attaquant peut n'avoir à disposition qu'un capteur normal (*mote-class attacks* en anglais), qu'il lui appartienne ou bien qu'il ait été compromis parmi les capteurs déployés à l'origine. Il se retrouve dans ce cas avec des machines similaires aux appareils attaqués, mais n'a pas besoin de rester sur place et peut mener des attaques sur la durée. Il peut également être équipé d'un ordinateur plus puissant (*laptop-class attacks*), voir de matériel militaire spécialisé. Dans ce cas, il est plus aisé de déployer de la puissance (puisque l'attaquant peut s'affranchir des limites imposées par les batteries des capteurs) et par exemple de brouiller en continu toute une plage de fréquence. Il faut néanmoins que le matériel utilisé reste déployé le temps de mener l'attaque, ce qui peut s'avérer couteux sur la durée.

2.1.4 ▶ Attaques actives, passives

Cette méthode de classement ne s'applique pas ici. Les attaques dites « passives » n'interfèrent pas avec le fonctionnement normal du réseau : il peut s'agir par exemple d'écoute clandestine en vue de collecter des données (atteinte à la confidentialité des communications), mais par définition les attaques de déni de service sont des attaques dites « actives », au cours desquelles l'attaquant introduit de nouveaux comportements dans le réseau [120].

2.1.5 ▶ Selon le paradigme considéré

Les méthodes de classification présentées jusqu'à maintenant restent limitées : elles consistent le plus souvent à établir une simple dissociation des catégories sur un nombre très restreint de critères. Les méthodes abordées à présent sont plus générales, et permettent des catégorisations plus représentatives, plus précises des attaques.

Certaines études se penchent sur une classe d'opérations réalisées par les réseaux de capteurs, et répertorient les attaques et les contre-mesures qui s'appliquent à cette classe [69, 96]. Les paradigmes suivants peuvent être considérés :

- la collecte et la transmission simples des données (paradigme qui se concentre sur une collecte simple et un envoi direct à la station de base, sans traitement, sans routage dans le réseau) ;
- la transmission des données d'un point à un autre du réseau (c'est-à-dire tout ce qui touche au routage des paquets dans le réseau, donc en faisant intervenir la retransmission des paquets par plusieurs nœuds intermédiaires) ;
- la réception et le traitement de commandes (dans le cas où les nœuds sont susceptibles de communiquer entre eux et de s'échanger des commandes, pouvant mener à des changement de configuration des capteurs) ;
- l'organisation autonome du réseau (les réseaux de capteurs s'organisent de façon autonome, mais des attaques peuvent chercher à interférer avec la formation d'une architecture cohérente ; dans ce paradigme sont inclus les protocoles de clusterisation éventuellement mis en application) ;
- l'agrégation de données (qui consiste à agréger, éventuellement compresser les données reçues avant retransmission, dans le but de limiter la taille et le nombre de paquets envoyés, afin de minimiser l'utilisation du canal de transmission, et surtout la consommation en énergie des capteurs) ;
- l'optimisation du modèle utilisé (qui intervient lorsque les capteurs ont des décisions à prendre, basées sur le contenu des paquets, telles que la retransmission directe ou différée, le niveau de sécurité à fournir…).

Cette méthode de classement est la plus utile pour se concentrer sur un point particulier du fonctionnement d'un réseau, et relever toutes les failles susceptibles d'affecter les opérations concernées.

À noter qu'il existe dans la littérature des classifications ontologiques (le système est représenté sous forme de graphe relationnel) des attaques par déni de service [129], qui se rapproche quelque peu de la méthode présentée ici, même si les exemples traités portent le plus souvent sur les réseaux en général sans évoquer les capteurs.

La table 3.4 située plus bas présente un classement selon les paradigmes des attaques que nous allons introduire.

2.1.6 ▸ Selon les couches de protocoles concernées

Une seconde méthode efficace de classement consiste à procéder par couches de protocoles, en se basant sur le modèle TCP/IP [120]. Ce modèle est rappelé en figure 3.1.

		Exemples :
5	Application	HTTP, FTP, SSH
4	Transport	TCP, UDP
3	Réseau	IP
2	Liaison	IEEE 802.11 (CSMA/CA)
1	Physique	ondes électromagnétiques

FIG. 3.1 : Modèle TCP/IP (rappel)

Ce classement permet une revue efficace, couche par couche, de la plupart des attaques connues. C'est donc selon ce critère que nous allons maintenant présenter les principales attaques par déni de service connues dans les réseaux de capteurs.

2.2 Différents types d'attaques

2.2.1 ▸ Couche physique

La couche physique du réseau correspond au médium physique employé pour la transmission des données entre deux nœuds, et à la façon dont le signal est transmis au travers de ce médium. Dans le cas de réseaux sans fil, le signal est propagé sous forme d'ondes électromagnétiques qui se déplacent dans le vide (ou bien, sans en être affectées, à travers l'atmosphère). Sauf s'il est fait usage d'une antenne directionnelle pour l'émission, ces ondes sont envoyées dans toutes les directions, et tout appareil à portée équipé d'un récepteur se retrouve donc en mesure de recevoir les paquets émis.

Brouillage de fréquences Le brouillage de fréquences (*frequency jamming* en anglais) consiste pour l'attaquant à émettre un signal parasite, un « bruit » électromagnétique, sur les fréquences concernées, de façon à ce que la cible visée ne puisse plus recevoir de façon correcte les paquets qui lui sont envoyés par les nœuds légitimes [98]. Le brouillage peut être réalisé à l'aide d'une antenne directionnelle pour viser une cible en particulier ; mais dans le cas d'un réseau de capteurs, l'attaquant cherche en général à émettre un bruit dans toutes les directions afin d'affecter le plus grand nombre de nœuds possible. L'attaque peut être menée de façon sporadique, de manière à produire un déni de service partiel, ou bien en continu. Si la ou les machine(s) émettant le signal parasite possède(nt) une portée suffisante pour couvrir toute l'étendue géographique du réseau, l'intégralité des capteurs peuvent se retrouver dans l'impossibilité d'utiliser les fréquences brouillées. Si de plus, le brouillage est mené sur toute la plage de fréquences accessibles aux capteurs, les communications deviennent totalement impossibles à établir dans le réseau.

Il est à noter que mener une telle attaque peut être couteux en matériel pour l'attaquant, surtout s'il vise plusieurs fréquences et/ou un brouillage continu dans le temps. Typiquement, un capteur corrompu ne sera pas en mesure de mener cette attaque sans épuiser très rapidement sa batterie.

2.2.2 ▶ Couche liaison de données

La couche de liaison de données fournit les moyens fonctionnels et procéduraux pour le transfert des données entre deux entités du réseau. Elle permet aussi, le plus souvent, de détecter et éventuellement corriger certaines erreurs survenues sur la couche physique (en cas de perturbation ou dégradation du signal électromagnétique) [121]. Des deux sous-couches LLC et MAC, c'est principalement la seconde qui nous intéresse ici : le protocole utilisé à ce niveau définit la manière dont les différents agents du réseau accèdent au médium de transmission de façon à limiter les collisions, et à garantir un accès le plus souvent équivalent au médium pour tous les nœuds. Les différents modes d'accès au médium existants ont été résumés au chapitre 2, sous-section 1.4 : certains consistent à créer des canaux de transmission distincts, tandis que d'autres déterminent l'accès à une même bande de fréquences en instaurant des règles. Ces règles peuvent être contournées, et la couche MAC va donc se retrouver associée à plusieurs types d'attaques.

Création de collisions et brouillage « intelligent » Lorsque plusieurs nœuds dont les portées se chevauchent émettent de façon simultanée en utilisant la même fréquence (donc sur un même canal), il se produit des collisions. La plupart des protocoles MAC employés avec les réseaux de capteurs introduisent dans les trames un champ contenant une somme de contrôle, qui permet de vérifier l'intégrité de la trame. Mais cette somme de contrôle ne permet pas, la plupart du temps, de corriger d'éventuelles erreurs (aucun des standards IEEE 802.11 (Wi-Fi), IEEE 802.15.1 (Bluetooth) ou IEEE 802.15.4 (ZigBee, 6LoWPAN) n'inclue de code de correction des erreurs). Si un seul bit de la trame est altéré, celle-ci est donc rejetée par le destinataire. Un attaquant peut donc chercher à produire des collisions en émettant un signal en même temps qu'un nœud légitime, afin que le destinataire ne puisse pas recevoir correctement la trame qui lui est destinée. Ce principe de collision est identique au brouillage mené sur la couche physique ; mais lorsque l'attaquant a connaissance du protocole de couche MAC employé, il lui est possible d'affiner son attaque, et de remplacer un brouillage « brut », continu et onéreux, par un brouillage « intelligent ». Il existe plusieurs façons de procéder [98] :

- la méthode de base portant sur la couche physique, comme vu plus haut, n'est pas « intelligente » et consiste à brouiller de façon continue les fréquences radio, en produisant un bruit aléatoire [98] ;
- une variante consiste à brouiller les fréquences sur des intervalles de temps distincts (et aléatoires), toujours à l'aide d'un bruit aléatoire (*random jamming*, « brouillage aléatoire » en anglais) : la réduction de la durée d'émission permet à l'attaquant d'économiser de l'énergie ;
- une technique plus difficile à détecter consiste à envoyer des paquets normaux mais de façon continue, de façon à occuper le canal sans interruption [98]. Elle empêche les nœuds légitimes de transmettre dans le cas ou le protocole de couche MAC est CSMA/CA par exemple (on parle en anglais de *deceptive jamming*, « brouillage trompeur ») ;
- une technique plus économe consiste à envoyer des bits à intervalles distincts (réguliers ou non), des *cybermines* en anglais, dans l'espoir de créer des collisions [98]. Il n'est alors plus nécessaire d'émettre en continu, et l'attaque est

moins facile à détecter. Elle est bien sûr moins efficace, car l'attaquant n'a plus la garantie de brouiller tous les paquets émis ;

- la méthode dite *reactive jamming* (« brouillage réactif ») consiste à ne pas transmettre de façon aléatoire, mais uniquement lorsqu'une émission émanant d'un autre nœud est détectée[98]. Elle permet une meilleure conservation de l'énergie, mais ne joue plus sur les mécanismes d'esquive de collision, qui font régulièrement reporter, avec les techniques précédentes, leurs transmissions aux nœuds légitimes dans l'attente de la libération du canal.
- les attaques « intelligentes » à proprement parler reposent sur la connaissance des spécifications du protocole MAC employé, et consistent à jouer sur la nature des paquets de contrôle et les délais d'attente [98]. Ainsi, pour le protocole CSMA/CA employé avec la suite IEEE 802.11 [61], une trame spécifique dite RTS (*Request To Send* en anglais) de demande de réservation du canal est suivie d'une trame CTS (*Clear To Send*) après une durée SIFS (*Short InterFrame Space*) si le destinataire est prêt à recevoir des données. En créant une collision sur la seule trame CTS, l'attaquant prévient entièrement l'échange de données utiles entre les nœuds. D'autres trames peuvent être utilisées dans le même but, leurs sigles sont dans la table 3.2.

Tab. 3.2 : Brouillage « intelligent » : corruption des trames de contrôle

Paquet analysé	Temps d'attente	Paquet à corrompre
RTS	SIFS	CTS
DATA	SIFS	ACK
RTS	DIFS	DATA
Quelconque	DIFS	RTS ou DATA

Épuisement de la batterie Corrompre des trames à moindre cout permet à un attaquant de forcer une cible à émettre davantage de trames que prévu, puisqu'un échec de communication est généralement suivi de plusieurs autres tentatives. À chaque nouvel essai, la cible doit émettre à nouveau la trame complète, et puise donc dans ses réserves énergétiques pour alimenter son bloc d'émission. Étant donné qu'il est très difficile de changer la batterie d'un capteur une fois le réseau déployé, un capteur dont la batterie est vide cesse d'être opérationnel, et ne remplit plus du tout son rôle. L'attaquant peut donc chercher à épuiser la batterie d'un ou de plusieurs nœuds à portée, soit en leur faisant émettre plus de trames que nécessaires (collisions), soit en les retenant le plus possible en état d'activité (émission ou écoute du canal) pour les empêcher de rentrer dans un état de veille qui permet l'économie des ressources (attaque par privation de sommeil, ou *denial of sleep* en anglais) [86]. Une autre méthode consiste encore à pousser la cible à mener des calculs intensifs : lorsque des algorithmes cryptographiques sont mis en application notamment, chaque requête, même rejetée, se traduit par un nombre important de calculs nécessaires pour vérifier la signature d'un message. Si l'attaquant se sert d'un capteur corrompu ou d'un appareil équivalent pour mener son attaque, il lui est essentiel d'émettre le moins de signaux possibles pour économiser son énergie, et ne pas vider sa batterie au même rythme que ses cibles (au risque de devenir inactif avant les nœuds visés).

Accaparement du canal de transmission Un appareil compromis introduit dans le réseau peut chercher à s'octroyer un accès plus important que les agents légitimes au canal de transmission. Ce comportement cupide (ou *greedy* en anglais) peut être atteint par exemple en jouant sur les paramètres du protocole de couche MAC [98] : non respect des durées minimales DIFS entre deux envois (pour IEEE 802.11 par exemple), non respect des valeurs employées pour la taille des fenêtres de congestion, non respect de la réservation des ressources par d'autres nœuds, *et cætera*. Le but final de telles attaques peut être d'envoyer plus de données à la station de base, dans le but par exemple de fausser une moyenne, ou bien de transmettre les données écoutées (attaque sur la confidentialité) dans le voisinage à une autre machine corrompue du réseau, en vue de l'exfiltration de ces données utiles capturées. On parle aussi parfois en anglais d'*unfairness* (« injustice ») lorsque les spécifications des protocoles mis en place sont délibérément ignorées par un nœud corrompu.

Falsification d'accusés de réception (*ACK spoofing*) La falsification d'un accusé de réception (d'un paquet ACK, pour *acknowledgement*, « accusé de réception » en anglais), à destination d'un nœud venant d'émettre des données, permet à un attaquant d'empêcher ce nœud de réaliser d'autres tentatives de transmission de la trame [7]. Elle n'est utile, bien entendu, que dans le cas où l'attaquant pense que l'émission de la trame d'origine a échoué, par exemple parce qu'il a créé une collision sur le canal lors de cette première émission. C'est une attaque qui vient compléter les techniques de brouillage de façon à empêcher la transmission de l'information dans le réseau.

2.2.3 ▶ Couche réseau

Le protocole IP est le plus employé sur la couche réseau dans les réseaux de capteurs, pour assurer l'adressage des paquets indispensable à la mise en place d'un algorithme de routage qui détermine comment ces derniers sont retransmis saut après saut dans le réseau. Dans le cas d'un réseau clusterisé, il arrive que tous les capteurs soient à portée directe de leur cluster head, et que ce dernier soit en mesure d'atteindre directement la station de base. Le routage est alors très simple. Mais dans d'autres cas, il est nécessaire d'établir une structure pour le réseau permettant l'acheminement des paquets jusqu'à leur destinataire. Un protocole de routage efficace doit minimiser les pertes de paquets ainsi que les couts de retransmission et éviter la création de boucles dans le réseau. À l'inverse, un attaquant peut tenter de mener une attaque par déni de service en gênant le plus possible l'acheminement de ces paquets.

Trou noir (*blackhole*) Une attaque de type « trou noir » (ou *blackhole* en anglais) consiste pour un nœud compromis à n'effectuer aucune retransmission des paquets qui lui sont envoyés. Le nœud accuse réception du paquet auprès de l'émetteur, mais ne transmet jamais le paquet pour le prochain saut prévu pour ce paquet par le protocole de routage. Toutes les routes qui passent par ce nœud compromis mènent donc à la suppression pure et simples des paquets en cours de transit [70].

Retransmission sélective de paquets (trous gris (*grey holes*), attaques « on/off » (*on/off holes*)) Là où une attaque de type « trou noir » supprime tous les paquets au lieu de les retransmettre, il est possible de ne retransmettre à la place qu'un sous-ensemble de paquets. L'attaque devient plus difficile à détecter, puisque le nœud compromis retransmet tout de même des paquets de temps à autre.

Les attaques basées sur ce mécanisme sont dites à retransmission « on/off » (*on/off holes* en anglais) lorsque la retransmission de tous les paquets est soit complètement assurée, soit complètement suspendue en alternant entre ces deux phases au fil du temps. On parlera plutôt d'attaques de type « trou gris » (*grey holes* en anglais) lorsque la retransmission est partielle mais constante dans le temps, qu'elle soit basée sur un mécanisme aléatoire (paquets supprimés au hasard plutôt que d'être retransmis) ou selon des règles (selon le contenu utile, ou selon l'expéditeur ou le destinataire par exemple) [138].

Falsification des informations de routage (accaparement du canal de transmission, création de boucles, *et cætera*) Une attaque peut être menée dès le déploiement du réseau en transmettant de fausses informations lors de la mise en place des règles de routage [7]. Ces informations vont alors chercher à nuire au routage des paquets, en créant des boucles cycliques dans la structure de routage, en partitionnant le réseau, en attirant à tort le trafic vers un nœud corrompu (par exemple pour exploiter les données et porter une attaque sur la confidentialité) ou vers un nœud légitime (pour surcharger ses capacités de traitement)…

De telles attaques peuvent aussi être menées lorsqu'un protocole de clusterisation est mis en application, dans le but de gêner l'organisation structurelle du réseau (puisque la mise en place des clusters définit le plus souvent la façon dont les paquets seront routés dans le réseau).

Plusieurs autres attaques présentées ci-dessous reposent sur la falsification des informations de routage.

Puits (*sinkhole*) Une attaque *sinkhole* en anglais est la combinaison d'une attaque de type « trou noir » avec la diffusion de fausses informations de routage, en vue d'attirer le maximum de paquets possible vers le nœud attaquant [32]. Concrètement, un nœud corrompu peut se déclarer voisin direct de la station de base et annoncer une route de cout nul vers cette dernière. Ses voisins vont estimer qu'il s'agit du plus court chemin pour acheminer les paquets jusqu'à la station de base, et vont transmettre l'information à leur tour. Au final une part importante des routes créées, et en conséquence du trafic routé, va passer par ce nœud compromis, ce qui peut mener à des congestions dans le réseau. Rentre alors en jeu l'attaque *blackhole*, qui supprime tous les paquets reçus plutôt que de les retransmettre à leur destinataire légitime.

Une conséquence supplémentaire de cette attaque est l'épuisement de la batterie des nœuds voisins de l'attaquant : comme celui-ci se déclare très proche de la station de base, un grand nombre de routes vont rediriger les paquets vers lui, et ses voisins vont donc se retrouver fortement sollicités par des nœuds plus éloignés pour lui acheminer des paquets.

Il est à noter que la littérature inverse parfois, et même confond de temps en temps, la terminologie des attaques de type *blackhole* et *sinkhole*. Les définitions présentées ici sont celles qui semblent revenir le plus couramment [1]. Par ailleurs, tous les « trous » ne font pas nécessairement référence à des pertes de paquets : des trous physiques, logiques ou sémantiques peuvent se former dès le déploiement du réseau [4, 67], mais

[1] NdA : Cette terminologie est donc contraire à l'image que l'on pourrait se faire d'un trou noir qui attire à lui la matière environnante sous l'effet de la gravité, tandis qu'un simple puits n'attire rien. En l'état actuel je suppose, sans avoir réussi à le vérifier, que le nom de l'attaque *sinkhole* provient de la station de base, souvent appelée « puits » ou *sink*, dont le nœud compromis se fait passer pour un voisin immédiat lors de l'attaque, et qu'il « remplace » en quelque sorte.

il s'agit d'un problème qui tient davantage de la sureté et de la résilience que de la sécurité.

Trou de ver (*wormhole*) Lorsque deux agents ou plus du réseau sont compromis par un attaquant, il leur est possible de mener une attaque de type « trou de ver » (*wormhole attack* en anglais). Cette attaque consiste à capturer du trafic en un point donné du réseau pour le réinjecter en un autre point. Il faut donc au moins deux agents complices, l'un qui capture et l'autre qui injecte le trafic, et qui communiquent l'un avec l'autre par le biais d'un canal auxiliaire généralement distinct des canaux légitimes utilisés sur le réseau (un tunnel, ou « trou de ver », qui donne son nom à l'attaque) [38]. Il est intéressant, pour les nœuds menant l'attaque, de retransmettre par exemple les informations (de routage, notamment) provenant du voisinage de la station de base en un point éloigné du réseau, de façon à faire passer le tunnel pour une route de bonne qualité vers la station de base, puis de mener des attaques de retransmission sélective. Ou plus simplement, l'injection en un autre point des paquets utilisés pour la découverte de routes lors de la mise en place de la structure de routage peut nuire grandement à l'organisation du réseau, et conduire un nœud légitime à imaginer des voisins virtuels qui sont en réalité absolument hors de sa portée d'émission.

Déluge de paquets « hello » (*"hello" flooding*) Un mécanisme souvent mis en place dans les réseaux de capteurs est l'envoi de paquets de type « hello » pour découvrir quels sont les nœuds dans leur voisinage. À la réception d'un tel paquet, les nœuds voisins émettent en réponse un paquet « hello-reply » indiquant au premier capteur qu'ils ont reçu le message. Un attaquant, s'il dispose d'une machine plus puissante que la moyenne des capteurs, peut forger et envoyer des paquets « hello-reply » avec une portée supérieure à celle des capteurs, pour leur faire enregistrer des voisins qui sont en réalité bien au-delà de leur portée d'émission, et désorganiser complètement le routage de paquets dans le réseau [7].

Altération des données Lorsqu'aucune vérification de l'intégrité des données n'est réalisée, un nœud relai corrompu peut, s'il a accès au contenu (en clair ou parce qu'il possède la clé pour déchiffrer), modifier ou retrancher des données du paquet à retransmettre [120]. Les données qui parviennent à l'exploitant du réseau sont alors erronées ou incomplètes.

Attaque attaque Sybil Le nom de l'attaque « Sybil » provient du titre du roman éponyme écrit par Flora Rheta Schreiber, publié en 1973 et racontant le traitement d'une patiente souffrant de multiples dédoublements de la personnalité. L'attaque en elle-même consiste justement pour un agent corrompu à endosser l'identité de plusieurs nœuds dans le réseau. Ce peut être l'identité :

- inventée, de nœuds inexistants ;
- de nœuds existants, mais distants du nœud corrompu ;
- de nœuds détruits et virtuellement remplacés par le nœud corrompu.

Cette mascarade permet au nœud compromis de faire échouer d'éventuels schémas de résilience, par exemple lorsque le protocole de routage prévoit des routes « de secours » si les routes principales venaient à être coupées. Des schémas de partition,

de réplication, de distribution du stockage de l'information se retrouvent également affectés si des nœuds qui sont considérés comme distincts par les agents légitimes sont en fait simulés par un unique capteur compromis. Le mécanisme peut également être utilisé dans le but d'intensifier d'autres attaques [94], notamment :

- l'altération des données finales récupérées par l'exploitant, en transmettant des données erronées en provenance d'un grand nombre de capteurs virtuels ;
- l'exclusion de nœuds légitimes du réseau dans le cas où une solution de détection est mise en place (voir plus bas), en simulant un comportement irrégulier de la part de ces nœuds, ou en votant contre eux sous le couvert de plusieurs identités de façon à remporter ces votes ;
- l'accaparement de ressources, en réservant des ressources (créneaux temporels pour l'accès au canal…) pour plusieurs nœuds virtuels, mais qui ne seront utilisées que par le seul capteur menant l'attaque.

De manière générale, en l'absence d'un système efficace d'authentification, les attaques reposant sur des techniques d'usurpation d'identité peuvent être très compliquées à détecter et à contrecarrer correctement.

2.2.4 ▶ Couche transport

Les protocoles de couche transport ne sont pas toujours implémentés dans les réseaux de capteurs sans fil, mais lorsqu'ils sont présents, des attaques peuvent profiter de leurs spécifications.

Déluge de paquets SYN (ou équivalents) (*SYN flooding*) Les attaques par déni de service existant sur les réseaux classiques au niveau de la couche transport peuvent aussi être appliquées dans les réseaux de capteurs : par exemple, si le protocole TCP est utilisé dans le réseau, un attaquant peut inonder le réseau de paquets SYN utilisés pour initier des connexions entre deux nœuds [7]. Cette attaque requiert une machine plus puissante (surtout, avec une meilleure alimentation en énergie) qu'un capteur, mais permet à la fois de créer des congestions dans le réseau, et de saturer les capacités des capteurs en ouvrant un nombre trop grand de sessions TCP.

Désynchronisation TCP (ou équivalents) Dans le même registre, un attaquant peut forger des demandes de désynchronisation pour mettre fin à des sessions TCP établies entre deux entités légitimes. Les échanges de ces sessions sont donc interrompus jusqu'au rétablissement d'une nouvelle connexion : ces connexions impliquent l'envoi de données de contrôle (elles s'effectuent en trois temps, on parle de *three-way handshake* en anglais) qui consomment une quantité d'énergie précieuse pour les capteurs.

De manière plus générale, si un protocole de transport est employé, TCP par exemple, alors les attaques de déni de service reposant sur ce protocole [62] se retrouvent applicables dans le réseau.

2.2.5 ▶ Couche application

La couche application implémente éventuellement l'application utilisée au niveau le plus haut par le réseau de capteurs pour assurer un service en particulier. Les protocoles utilisés sur cette couche dépendent donc totalement de l'objectif final du réseau,

il n'y a pas ici de standard à proprement parler. Certaines attaques sont néanmoins applicables au niveau applicatif.

Données erronées Un nœud compromis a la possibilité d'envoyer des données en parfaite contradiction avec les valeurs physiques mesurées (voire même, il peut s'affranchir des mesures et ainsi économiser de l'énergie). Les valeurs transmises au niveau application viendront alors fausser les résultats obtenus par l'exploitant du réseau [120].

Déluge de paquets Suivant l'application mise en place, il peut être envisageable pour un attaquant d'inonder le réseau de données utiles, qu'elles aient été effectivement mesurées ou non, soit dans une tentative de fausser les résultats obtenus par la station de base en faisant des moyennes sur les valeurs rapportées par l'ensemble des capteurs, soit pour créer des congestions dans le réseau [9].

Désynchronisation Si jamais le protocole utilisé sur la couche application induit la création de sessions en mode connecté (comme pour TCP sur la couche transport), des attaques de désynchronisation peuvent être menées par un attaquant en vue de briser ces sessions [7]. La désynchronisation peut aussi être menée sur l'horloge des nœuds, pour les empêcher d'établir le même référentiel temporel, ce qui peut nuire au fonctionnement correct de certaines applications [120].

2.2.6 ▶ Hors modèle

Destruction physique des capteurs Les capteurs sont des appareils de petite taille déployés en nombre, et dont la production ne doit par conséquent pas être trop couteuse. Cela implique que les appareils sont relativement fragiles, et d'autant plus difficiles à sécuriser que le réseau est souvent déployé en extérieur, parfois en environnement hostile. À distance, il est éventuellement possible de mettre des capteurs individuels hors service dans le cas où un nœud compromis parvient à faire épuiser la batterie [86], ou bien la mémoire disponible des capteurs [132]. Mais un attaquant peut parfois accéder directement au capteur lui-même [43]. C'est l'une des manières de s'emparer d'un capteur pour le reprogrammer et introduire un agent corrompu dans le réseau (l'autre méthode reposant sur des failles logicielles, qui permettent parfois l'injection distante de code). Mais sans s'embarrasser de les reprogrammer ou d'en épuiser les ressources, l'attaquant qui obtient un accès physique au réseau peut aussi se contenter de détruire physiquement les capteurs, ce qui est encore le moyen le plus efficace de les empêcher de fournir les services pour lesquels ils ont été conçus.

Altération des mesures Toujours en raison de leur exposition physique, les capteurs peuvent être manipulés par un attaquant pour mesurer des données. Des engins munis de caméras peuvent ainsi voir leur objectif obstrué (par une couche de peinture par exemple) ; ou bien des capteurs chargés de mesurer la température d'un élément donné peuvent être déplacés, de telle sorte que leur thermomètre ne soit plus en contact avec cet élément. Les capteurs continuent dans ce cas à mesurer des valeurs et à les retransmettre jusqu'à la station de base, mais ces mesures elles-mêmes (et donc les résultats obtenus par l'exploitant) se retrouvent faussées dès le départ.

Les tables 3.3 et 3.4 synthétisent le classement de ces différentes attaques, respectivement par couche du modèle TCP/IP et par paradigme considéré. Toutes les attaques exposées dans cette sous-section ont été décrites à maintes reprises dans la littérature, et notamment réunies au travers de plusieurs études portant sur l'état de l'art de la sécurité dans les réseaux de capteurs [116, 102, 7]

TAB. 3.3 : Classement des attaques par couche du modèle TCP/IP

COUCHE	ATTAQUES POSSIBLES
Hors modèle	1. Destruction physique des capteurs 2. Altération des mesures
Couche physique	3. Brouillage de fréquences
Couche liaison de données	4. Création de collisions, brouillage « intelligent » 5. Épuisement de la batterie 6. Accaparement du canal de transmission 7. Falsification d'accusés de réception
Couche réseau	8. Trou noir 9. Retransmission sélective de paquets (trous gris, attaques « on/off ») 10. Falsification des informations de routage (accaparement du canal, création de boucles, *et cætera*) 11. Puits 12. Trou de ver 13. Déluge de paquets « hello » 14. Altération des données 15. Attaque Sybil
Couche transport	16. Déluge TCP (ou équivalents) 17. Désynchronisation TCP (ou équivalents)
Couche application	18. Données erronées 19. Déluge de paquets 20. Désynchronisation

Il est bien sûr très difficile de prétendre à l'exhaustivité, et il existe d'autres attaques par déni de service applicables aux réseaux de capteurs : certaines dépendent d'applications particulières ou sont introduites par des configurations précises (mobilité des nœuds, structures spécifiques : grille, recherche de couverture maximale…). D'autres n'ont sans doute pas encore été découvertes. Les attaques que nous avons présentées regroupent néanmoins les principales menaces, en matière de déni de service, auxquelles peuvent avoir à faire face les réseaux de capteurs.

2.2.7 ▶ Discussion : plausibilité des attaques

Toutes les attaques que nous avons présentées dans cette sous-section sont abondamment décrites dans la littérature. Mais parmi elles, toutes ne rencontreront pas forcément le même « succès » dans le monde réel, et l'on peut s'interroger : quelles sont les attaques les plus plausibles, c'est-à-dire celles qui ont le plus de chances d'être implémentées par un attaquant réel ?

Il est bien sûr très difficile de se procurer des données à ce sujet, puisque lorsqu'une véritable attaque est menée, ses conséquences, et surtout son mode opératoire, ne

TAB. 3.4 : Classement des attaques par paradigme (voir sous-sous-section 2.1.5)

PARADIGME	ATTAQUES POSSIBLES
Collecte et transmission simples	Brouillage, création de collisions, destruction des capteurs, altération des mesures, altération des données (applicatif)
Routage des données dans le réseau	Retransmission sélective, trous noirs, épuisement des ressources (batterie, congestion du réseau), altération des données (lors de la retransmission), accaparement du canal de transmission, attaques au niveau de la couche transport
Réception et traitement de commandes	Attaques précédentes, usurpation d'identité (et attaque Sybil) utilisée pour émettre de fausses commandes
Organisation autonome du réseau	Attaques précédentes, fausses informations de routage (boucles…), puits, trous de ver, déluge de paquets « hello »
Agrégation de données	Attaques précédentes, en particulier celles basées sur le rejeu de paquets émis ou capturés
Optimisation des modèles	Attaques précédentes

sont pour ainsi dire jamais rendus publics. L'organisme qui gère le réseau de capteurs attaqué, surtout s'il fait partie du domaine militaire, n'a pas d'intérêt à communiquer sur les spécifications de son système, et encore moins sur ses vulnérabilités.

Cela ne nous empêche pas néanmoins de nous livrer à quelques conjectures.

Critères logiques Pour ne conserver que des attaques plausibles, nous pouvons tout d'abord éliminer celles qui se basent sur les failles de protocoles « obscurs », peu utilisés : si l'on trouve très peu d'implémentations de ces protocoles, très peu d'attaques qui leur sont associées seront déclenchées.

Nous pouvons également éliminer les attaques qui nécessitent de gros efforts de rétroingénierie pour analyser le fonctionnement précis de protocoles ou de systèmes « maison », car cette analyse détaillée coute cher à l'attaquant.

De même, si l'attaque requiert un équipement matériel onéreux, il faudra que l'attaquant soit prêt à investir. Ce n'est pas toujours un problème, surtout dans le domaine militaire ; mais s'il existe d'autres types d'attaques moins couteuses permettant d'arriver à un résultat similaire, autant limiter les dépenses.

Enfin, les attaques les moins « efficaces », celles qui n'ont que peu de chances d'aboutir, sont également à écarter.

Motivations de l'attaquant Nous avons présenté plus haut les conséquences directes sur le réseau que pouvait rechercher un attaquant, analysons maintenant l'objectif final qui le pousse à mener son attaque. Voici nos suppositions concernant les principales motivations des attaquants.

- **Neutraliser les communications** : l'attaquant souhaite mettre le réseau hors service. Il veut empêcher l'exploitant du réseau d'accéder aux informations collectées, que ce soit pour des raisons financières ou militaires. Il ne veut pas de communication entre les nœuds, et surtout, pas de remontées d'informations vers la station de base.

- **Détruire définitivement le réseau** : l'existence en soi du réseau est une nuisance pour l'attaquant. Celui-ci souhaite désactiver le réseau de façon définitive, car

il ne peut pas se permettre de le mettre simplement hors service de façon ponctuelle lorsque le besoin s'en ferait sentir.

- **Parasiter la collecte de donnés** : l'attaquant souhaite profiter du réseau mis en place par un concurrent pour obtenir des informations stratégiques à moindre frais. Il récupère et exporte des données circulant sur le réseau. Bien que cela ne soit pas une attaque de déni de service en soi, l'usage d'attaques par trous de ver, par accaparement du médium peuvent être menées pour rendre la tâche plus aisée, sans compter que la circulation supplémentaire des données exfiltrées risque de puiser davantage dans la batterie des capteurs légitimes.
- **Altérer les résultats** : l'attaquant souhaite induire sa cible en erreur en lui faisant parvenir des résultats erronés. Les prises de décision de l'opérateur vont alors se retrouver basées sur une vision erronée de la situation en cours. C'est l'exemple classique des séries d'espionnage, dans lesquelles les agents infiltrés font en sorte d'afficher l'image d'une pièce vide dans la salle de contrôle des dispositifs de vidéosurveillance afin de camoufler leur intrusion.

Les attaques correspondantes

Neutraliser les communications Le moyen le plus simple de mener une attaque de déni de service contre un réseau est encore de brouiller son signal, au niveau physique. La mise en place peut être couteuse en matériel, mais elle est très facile à mette en place. Menées de l'extérieur du réseau, elle ne requiert aucune connaissance sur le fonctionnement interne de celui-ci. L'attaquant n'a besoin de connaitre que la plage de fréquences à neutraliser, et à calculer la puissance d'émission nécessaire. Pour ces raisons, il s'agit donc de l'une des méthodes les plus utilisées dans le domaine militaire.

Les attaques de brouillage « intelligent » au niveau de la couche de liaison de données nécessitent d'écouter les trames émises dans le réseau, pour savoir à quel moment déclencher des collisions. Ces attaques peuvent être menées sur des protocoles simples et largement répandus, mais il serait long en revanche de les adapter à des protocoles moins « standards ». Elles peuvent également être menées depuis l'extérieur du réseau, ce qui évite d'avoir à compromette un agent. Cet avantage est important, car détourner un capteur de son but d'origine peut être très couteux en temps.

Aussi le brouillage, attaque externe, semble préférable, car plus simple, à bon nombre d'attaques plus complexes à mettre en place et nécessitant un accès dans le réseau. Il semble donc plus rentable que les attaques portant sur les protocoles de routage, ou bien les attaques de couche supérieure (par désynchronisation ou création de congestions), elles aussi destinées à rendre le réseau hors service.

Détruire le réseau Les deux façons principales de détruire le réseau sont la destruction physique des capteurs, et les attaques visant à l'épuisement des batteries. La destruction physique nécessite de connaitre l'emplacement des capteurs. Si ceux-ci sont relativement peu nombreux et assez faciles d'accès, il est possible de les retrouver par triangulation du signal et de procéder directement à leur mise hors service. S'ils sont très nombreux, ou s'il est compliqué de les atteindre (dans une zone naturellement ou militairement hostile par exemple), cette opération peut être très longue, voire impossible à mener. Une attaque par épuisement de batterie menée de façon

efficace à l'aide d'une poignée de nœuds compromis peut s'avérer plus rentable que d'aller détruire physiquement les nœuds, ou même que de brouiller le signal sur une très longue période. Il est nécessaire en revanche d'acquérir une connaissance suffisamment détaillée sur le fonctionnement interne du réseau pour mener l'attaque.

Récupérer des données L'exfiltration des données récoltées hors du réseau peut être menée de diverses façons, il n'est pas évident de déterminer si elle pénalisera le bon fonctionnement du réseau. En revanche, la collecte parasite elle-même peut passer par des attaques sur le protocole de routage, de façon à détourner le plus de données utiles vers le nœud corrompu. L'annonce de fausses routes, typiques des attaques *sinkhole*, se prêtent particulièrement bien à la collecte frauduleuse de paquets (d'ailleurs, une fois ces paquets collectés par l'attaquant, à quoi bon les retransmettre ?).

En comparaison, les attaques de type *blackhole* ou *grey hole* se limitent à la suppression de paquets, sans influer sur la création des routes. Elles peuvent être utilisées pour neutraliser la remontée d'informations, mais une fois encore, on leur préfèrera dans ce contexte des attaques de brouillage.

Altérer les mesures Les attaques portant sur l'intégrité des données sur les différentes couches semblent efficaces pour altérer les résultats perçus par l'exploitant. Il y a toutefois une limite relative à l'application pour laquelle le réseau a été déployé : si les résultats finaux sont similaires à des moyennes calculées sur toute la zone couverte par les capteurs, influer significativement sur cette moyenne va demander d'altérer un grand nombre de paquets. L'envoi de messages en déluge pour augmenter artificiellement le nombre de valeurs et fausser les résultats ne semble pas très intéressant, car le procédé est extrêmement facile à détecter dès lors qu'un mécanisme d'authentification a été mis en place et que l'attaquant ne peut pas masquer son identité. La modification des paquets peut être menée sur un grand nombre d'entre eux pour peu que le nœud compromis soit bien situé, c'est-à-dire s'il s'agir d'un relai proche de la station de base.

Les attaques qui nous semblent les plus plausibles sont donc au final les suivantes :

- le brouillage radio, notamment au niveau physique, pour neutraliser les communications ;
- certaines attaques de routage et notamment les attaques de type *sinkhole*, pour pénaliser les communications mais aussi et surtout pour la collecte parasite de données ;
- les attaques par épuisement de la batterie, ou la destruction physique des capteurs, pour mettre le réseau hors service de manière définitive ;
- la corruption des données remontées à l'opérateur, pour lui fournir des résultats erronés et influencer sa prise de décisions.

Du point de vue de l'attaquant les techniques de brouillage « intelligentes » peuvent nécessiter des ajustements en fonction du protocole MAC employé dans le réseau, et mieux vaut mener des tests pour s'assurer que les attaques fonctionnent correctement. Les autres attaques servant à neutraliser les communications sont beaucoup plus complexes à mettre en place, et requièrent des connaissances sur les mécanismes

internes du réseau. Il n'est pas évident de trouver un intérêt concret aux comportements cupides dans les réseaux de capteurs en dehors de la collecte parasite d'informations.

Il serait intéressant de pouvoir recouper ces hypothèses avec des statistiques issues de déploiements réels : savoir quelles sont les attaques les plus utilisées permettrait de déterminer dans quelle direction diriger les efforts pour la défense du réseau, et par conséquent comment axer la recherche de solutions. Mais encore une fois, l'évaluation de la plausibilité des attaques réalisée ici n'est que le produit de conjectures et n'a pas pu être vérifiée par une étude concrète. En particulier, on imagine sans mal que pour un but spécifique, un attaquant disposant de suffisamment de moyens prenne le temps d'implémenter une attaque très complexe, après avoir analysé en profondeur le fonctionnement du réseau ciblé ; l'intrusion se retrouverait alors plus efficace dans les dégâts infligés, et/ou plus difficile à détecter. L'actualité de ces dernières années en matière de sécurité informatique, principalement depuis la découverte du logiciel malveillant Stuxnet [37] en 2010, tend en effet à prouver que des organismes importants sont prêts à investir des moyens considérables dans la mise en place d'attaques très sophistiquées. Il n'est pas difficile d'imaginer que ce comportement puisse être appliqué dans le domaine des réseaux de capteurs si le besoin s'en fait ressentir.

2.3 Prévenir les attaques

2.3.1 ▸ Authentification et problématiques similaires

La sécurité est transversale à l'ensemble des domaines d'études en informatique, et la protection contre le déni de service… est elle-même liée aux autres problématiques de sécurité. Il faut entendre par là que si des failles du système compromettant l'authentification des agents ou l'intégrité des données peuvent également être exploitées d'une façon ou d'une autre pour nuire au bon fonctionnement du réseau, alors l'attaque en question vient également se classer parmi les attaques de déni de service.

De manière plus concrète, assurer correctement l'authentification des nœuds du réseau empêchera un attaquant de mener des attaques basées sur l'usurpation de l'identité d'un autre nœud. Les attaques par désynchronisation, ou la falsification d'accusés de réception par exemple, deviennent inapplicables. Si des mécanismes permettant de limiter le rejeu de paquets capturés sont rajoutés, l'ampleur des attaques Sybil ainsi que des attaques par trou de ver se verront fortement réduites. La protection de l'intégrité des données assure de son côté l'échec de toute tentative d'altération des valeurs transmises lors des opérations de routage.

Les mécanismes dédiés à la garantie de ces différentes propriétés, présentés en section 1, sont donc également utiles pour protéger un réseau de capteurs de certaines attaques de déni de service. On parle de mesures de prévention, dans le sens où il s'agit de méthodes mises en place pour empêcher une attaque de survenir. De nombreuses architectures de sécurité, qui apportent authentification, chiffrement, intégrité et protection contre le rejeu aux échanges, et parmi lesquelles une bonne partie des solutions présentées en sous-sous-section 1.3.3, se « vantent » ainsi d'atténuer, voire de prévenir totalement plusieurs types d'attaques de déni de service.

2.3.2 ▶ Des protocoles de routage spécifiques

Toujours dans le but de prévenir les attaques, il est possible d'utiliser des protocoles de routage spécifiquement étudiés pour limiter les risques. Ainsi, le protocole de routage AODV (*Ad hoc On-demand Distance Vector routing*), très utilisé dans les réseaux de capteurs, a fait l'objet de propositions destinées à améliorer sa résistance aux attaques de déni de service de type *blackhole* et *sinkhole* : une étude propose par exemple de faire vérifier par chaque nœud, lors de la création des routes, que ses voisins de deuxième degré sont bel et bien accessibles [34]. Ainsi pour chaque nœud x ayant un voisin n qui lui propose une route vers une destination donnée, x demande à n quel est le prochain saut $n+1$ sur cette route et tente de contacter ce nœud $n+1$ pour vérifier qu'il est effectivement accessible. Si $n+1$ répond, alors n est digne de confiance pour cette route ; puisqu'il est digne de confiance, on admet qu'il pourra à son tour vérifier l'accessibilité du nœud $n+2$ à travers le saut annoncé par $n+1$, et ainsi de suite tout au long de la route empruntée pour les données.

Toujours pour parer aux attaques sur le routage, d'autres méthodes se basent sur des mécanismes de redondance et de résilience, pour faire en sorte que si l'une des routes se révèle défectueuse, d'autres exemplaires des paquets, envoyés par des routes distinctes, puissent attendre leur destination. Des mécanismes tels que le partage de secret peuvent être utilisés à cette fin de redondance, pour limiter l'augmentation du volume de données envoyées [19].

2.3.3 ▶ Prévenir la compromission des capteurs

Compliquer l'accès aux capteurs Il est en fin de compte très difficile de cacher la localisation géographique d'un capteur à un attaquant déterminé. Des mécanismes alliant clusterisation et usage de fausses sources pour les messages peuvent être employés afin de compliquer la tâche à l'attaquant [43]. L'usage d'antennes directionnelles peut également complexifier la localisation des nœuds, mais n'est pas adapté à toutes les utilisations des réseaux de capteurs. Au final, un attaquant correctement équipé pourra toujours trouver un capteur en procédant par triangulation du signal électromagnétique.

Il peut alors devenir intéressant de cacher à l'attaquant quels sont les nœuds qui représenteraient le plus d'intérêt pour lui. Un exemple traité est le jeu des « chasseurs de panda » : des capteurs dispersés dans l'habitat naturel d'un panda suivent les déplacements de l'animal et les rapportent à une station de base. Mais des braconniers recherchent l'animal et tentent d'utiliser les données du réseau en vue de le localiser plus facilement. S'il est impossible de cacher le signal des capteurs et d'empêcher les attaquants d'accéder directement aux machines, il est en revanche possible de déployer des mécanismes pour empêcher les chasseurs de savoir quel est le capteur qui renvoie à un instant donné les informations sur la présence du panda dans son secteur. Un mécanisme pour disséminer l'information, basé sur le modèle des épidémies, a ainsi été proposé dans cet exemple [73]. Si l'on se concentre uniquement sur le déni de service, il est tout à fait possible d'imaginer l'utilisation de techniques similaires afin de dissimuler à l'attaquant quels sont les nœuds à compromettre pour infliger un maximum de dégâts au bon fonctionnement du réseau.

Une autre approche possible [58] est de considérer que tout capteur ayant cessé de donner des « signes de vie » pendant une période prolongée est devenu hors-service, et qu'à ce titre il n'est plus censé réintégrer le réseau à l'avenir. Cette période est par exemple déterminée à l'aide d'un algorithme SPRT (*Sequential Probability Radio Test*)

qui calcule la durée au-delà de laquelle l'absence de signal devient anormal. Si le nœud tentait malgré tout de reprendre son activité au-delà de la période, alors supposition est faite que le capteur a été retiré du réseau par un attaquant le temps de procéder à une modification de son code, et ce nœud se retrouve alors volontairement exclu du réseau par mesure de précaution.

Durcir la sécurité logicielle Si un attaquant parvient à accéder physiquement à un capteur, il devient très compliqué de l'empêcher d'accéder au contenu de la mémoire de l'appareil, et/ou de le reprogrammer selon ses besoins. Des méthodes de chiffrement de la mémoire (principalement pour la mémoire non volatile) existent, mais sont peu implémentés sur les capteurs, car ils augmentent significativement la consommation en ressources de la machine et diminuent ses performances. Des modifications malveillantes du code exécuté par un nœud peuvent parfois être détectées, comme nous le verrons lorsque nous traiterons des systèmes de détection d'intrusion machine en sous-sous-section 2.4.1.

Mais outre les accès physiques directs, un attaquant peut très bien tenter de compromettre à distance un capteur, en exploitant des failles logicielles. Des injections de code ont déjà été menées avec succès sur des capteurs, en utilisant par exemple des dépassements de tampons pour injecter une fausse pile d'exécution et mener une attaque dite *return-into-libc* [42]. La sécurité logicielle des capteurs doit donc être vérifiée avant même le déploiement du réseau. Les bonnes pratiques d'ordre générale en sécurité s'appliquent ici comme ailleurs : de bonnes pratiques de développement pour le système sont indispensables, de même que la désactivation des services non utilisés (qui de toute façon consommeraient inutilement de l'énergie). La sécurité du système peut éventuellement être spécifiquement renforcée, comme par exemple avec grsecurity pour des systèmes basés sur le noyau Linux [47].

2.4 Détecter les attaques : systèmes de détection d'intrusion

2.4.1 ▸ Différents systèmes de détection d'intrusion

La détection des attaques par déni de service passe par la mise en place d'un système spécifique, apte à collecter des indices permettant de déterminer si une attaque a lieu et, si possible, quelle en est l'origine. On parle de « système de détection d'intrusion », ou IDS pour *Intrusion Detection System*. Pour être précis, il convient de relever que ces systèmes sont utilisés pour la détection de tous types d'attaques, y compris lorsqu'il ne s'agit pas de déni de service. Les usages et les spécifications des IDS sont donc multiples : la figure 3.2 résume les différentes catégories existantes, que nous allons brièvement décrire. Il existe différentes façon de les distinguer.

Selon l'origine de l'attaque Un système de détection d'intrusion peut se concentrer sur les attaques provenant de l'**extérieur** du réseau, menées par un attaquant qui chercherait à brouiller les fréquences utilisées ou bien, justement, à pénétrer dans le réseau.

À l'inverse, d'autres systèmes vont surveiller l'**activité interne** du réseau, afin de détecter d'éventuels nœuds corrompus, soit qu'ils cherchent à accaparer les ressources pour leur propre compte, soit qu'ils cherchent à détruire les ressources du réseau.

FIG. 3.2 : Critères de classification pour les systèmes de détection d'intrusion

Selon l'objet de la surveillance Un IDS peut avoir pour tâche de surveiller l'**activité du réseau**, en analysant le trafic : le nombre de paquets transmis, leur origine ou leur destination, leur contenu, le taux de succès des transmissions sont autant d'indices à collecter pour déterminer si le réseau fonctionne dans des conditions normales.

En parallèle, d'autres systèmes peuvent être orientés vers la **sécurité de l'hôte** sur lequel ils sont exécutés. Leur but est de détecter d'éventuelles tentatives de compromission de la machine. Les opérations menées à cette fin peuvent comprendre :

- la vérification de l'intégrité du système (par exemple en calculant périodiquement un condensat des fichiers exécutables critiques, afin de les comparer avec le condensat de leur version d'origine) ;
- la surveillance des journaux concernant les tentatives de connexion, afin de détecter des échecs répétitifs d'accès au système ;
- la détection d'une activité inhabituelle au niveau de l'activité du processeur, des allocations mémoires, ou des entrées sorties, qui risqueraient de pénaliser le système d'exploitation en matière de performances et de consommation énergé-

tique. Cette surveillance peut être menée en temps réel, ou via la journalisation des évènements système.

On parle en anglais de NIDS (*Network Intrusion Detection System*) ou de HIDS (*Host based Intrusion Detection System*) pour les IDS orientés réseau ou machine, respectivement. De nombreux IDS cumulent à la fois des mécanismes de surveillance du réseau et du comportement de leurs voisins, et une surveillance des évènements système survenant sur leur hôte [22].

Selon le type d'intrusion à détecter Les IDS sont utilisés pour lutter contre tous types d'« intrusions » (ou d'attaques dans le sens plus général), pas seulement contre le déni de service [22], bien que ce soit le point qui nous préoccupe le plus dans cet ouvrage. Ainsi ils peuvent être utilisés afin de détecter :

- des **tentatives d'accès au réseau** de la part d'un attaquant externe (qui chercherait à passer par exemple par la compromission d'un nœud ou par l'obtention de matériel cryptographique) ; cette étape n'est souvent qu'un prélude à d'autres attaques (par déni de service notamment) ;
- un **accès à des ressources non autorisées**, lorsqu'un capteur tente par exemple de procéder à certaines opérations dans un cluster qui n'est pas le sien ;
- les **fuites de données**, qui peuvent être difficiles à détecter si l'attaque est passive (simple écoute des données circulant sur le réseau), mais peuvent être identifiées lors de l'extraction de ces données vers l'extérieur du réseau ;
- les **comportements cupides** provoquant le blocage des ressources, par accaparement ;
- les autres attaques par déni de service, c'est-à-dire les **comportements destructeurs**, visant à nuire au fonctionnement du réseau par l'anéantissement des ressources, qu'elles soient virtuelles (mise hors service des transmissions) ou bien physiques (épuisement de la batterie des capteurs).

Selon les méthodes employées pour la détection Les méthodes utilisées pour déterminer l'existence et l'identité d'un attaquant sont nombreuses, et se divisent en trois catégories : la détection **basée sur les anomalies**, la détection **basée sur des signatures d'attaques**, et la détection reposant **sur la vérification du respect des spécifications** [22]. Ces catégories, et les principales méthodes qu'elles regroupent, sont détaillées plus bas en sous-sous-section 2.4.2.

Selon l'infrastructure du réseau Si de nombreux IDS ne se préoccupent pas de savoir quelle est la configuration exacte du réseau, et sont élaborés sur des architectures « plates », **sur un seul niveau**, d'autres en revanche sont spécifiquement construits avec l'idée de tirer profit d'une **architecture clusterisée**. La partition du réseau en clusters va de fait permettre la mise en place d'algorithmes différents, le plus souvent distribués au sein des clusters [113].

Il est également intéressant de noter que la clusterisation en soi d'un réseau est un mécanisme efficace pour limiter la portée de la plupart des attaques de déni de service, dont les dommages seront en principe circonscrits au cluster dans lequel elles sont menées. Une exception concerne toutefois les attaques qui sont menées avec succès contre un cluster head (notamment lorsque celui-ci se voit compromis), puisque ce peut être alors le cluster entier qui se retrouve hors service.

Selon la distribution des calculs La détection passe par la collecte puis le traitement d'indices sur le fonctionnement des nœuds ou du réseau global [22]. Collecte comme traitement peuvent être réalisés par différentes entités ou associations d'entités, à savoir :

- le traitement, les calculs peuvent être **centralisés** et réalisés seulement par la station de base, à laquelle les capteurs font remonter les informations brutes relevées sur le fonctionnement du réseau. Dans certains cas extrêmes, même la collecte d'indices n'est réalisée que par la station de base ;
- chaque capteur peut implémenter son propre système de détection d'intrusion : « **chacun pour soi** », quitte à garder pour lui ses conclusions sur le statut des autres nœuds. À l'inverse de la solution précédente, chaque capteur réalise alors de bout en bout les opérations de surveillance ;
- la détection peut être menée par des **agents mobiles**, qui se déplacent de nœud en nœud dans un réseau de capteurs autrement statiques (ou peu mobiles en comparaison avec les agents de surveillance). Ces agents mobiles disposent en principe de ressources plus importantes que les capteurs ordinaires. Cette configuration ne peut bien sûr pas être déployée pour tous les usages des réseaux de capteurs ;
- la collecte d'informations peut être **distribuée entre les nœuds** situés sur un même niveau d'horizontalité dans l'architecture du réseau : cette configuration leur permet d'échanger des informations entre eux, notamment des indices de confiance, et de déclencher des observations coordonnées lorsqu'un nœud « suspecte » un pair d'être compromis, sans en avoir la « certitude ». Lorsqu'un capteur est jugé définitivement compromis, l'information est partagée entre les nœuds. Dans certains systèmes, les calculs eux-mêmes peuvent être distribués ;
- enfin, la distribution peut également être réalisée de façon **hiérarchique**, sur un plan vertical, dans les réseaux clusterisés. Il y a alors interaction entre les membres d'un cluster et leur cluster head, qui se charge au minimum de la propagation des alertes, et parfois également de mener les calculs.

La gestion de la collecte et des calculs vont bien évidemment influencer l'opérateur du réseau sur le choix du type de méthode de détection (voir plus haut) à mettre en place.

Selon la fréquence d'usage du système La plupart des IDS sont prévus pour fonctionner **de manière continue** sur leur hôte, mais certaines solutions adoptent des comportement différents : le système de détection n'est exécuté que **de façon périodique**, ou bien se voit activé à la réception d'alerte (émanant de la station de base par exemple, qui dispose pour sa part des ressources nécessaires au maintien en continu d'un IDS) [22].

Systèmes hybrides En pratique, pour un critère donné, la plupart des IDS proposés se retrouvent dans plusieurs catégories à la fois : on retrouve ainsi beaucoup de systèmes hybrides surveillant tout à la fois l'hôte sur lequel ils tournent, et le trafic intercepté par le nœud. De même, les types d'attaques recherchés par les systèmes sont généralement multiples, et portent souvent sur plusieurs couches protocolaires (en référence au modèle TCP/IP). Et pour en détecter un maximum, il n'est pas rare

qu'un IDS combine plusieurs méthodes de détection. On voit ainsi souvent apparaitre en anglais le terme *cross-layer IDS*, littéralement « IDS sur plusieurs couches » [22].

2.4.2 ▶ Les méthodes de détection

Une fois le réseau déployé, et le système de détection d'intrusion mis en place, les nœuds chargés de la surveillance du réseau analysent le trafic environnant et en tirent des déductions sur l'état du réseau. Ces déductions résultent de l'application d'algorithmes précis sur les données. Ces algorithmes sont de plusieurs sortes, que l'on regroupe dans les méthodes de détection basées sur des anomalies, sur des signatures d'attaques, ou sur le non-respect de spécifications prédéfinies.

Détection basée sur les anomalies Détecter les attaques selon les anomalies qui surviennent dans le réseau revient à établir, dans un premier temps, un modèle du fonctionnement normal du réseau [22]. Dans un second temps, la surveillance en elle-même est déclenchée, et si les variables observées dérogent sensiblement au modèle généré, une alerte est levée. Ces méthodes permettent donc d'intervenir lorsqu'un problème survient effectivement dans le réseau, sans que son origine n'ait nécessairement été identifiée. Elles ont pour avantage de s'adapter à de nouveaux types d'attaques, pour lesquelles elles n'avaient pas été imaginées au départ. En revanche, elles peuvent être amenées à manquer certaines attaques connues dont les conséquences sont suffisamment discrètes. Un second inconvénient existe : elles requièrent un réajustement constant du modèle par rapport à l'évolution normale du réseau, et la surveillance doit donc être maintenue tout à la fois pour détecter les attaques et pour mettre à jour le modèle.

La détection basée sur les anomalies regroupe trois sous-catégories.

- La première est celle des **modèles statistiques**, qui enregistrent dans un premier temps des séries de valeurs pour des variables liées au bon fonctionnement du réseau, et cherchent par la suite à repérer les écarts importants à ces valeurs de référence. Les variables observées, directement liées au trafic, peuvent être considérées comme des variables indépendantes suivant des lois de Gauss, comme des variables corrélées, ou encore comme des séries temporelles.

- Une deuxième regroupe les modèles basés sur les **connaissances** modélisées du fonctionnement normal du réseau, et utilise un modèle généré en fonction des données et à l'aide d'outils tels que des langages de description ou des automates finis.

- Enfin, la détection peut être basée sur des **méthodes d'apprentissage**, où des modèles sont créés et actualisés *via* des outils formels comme les processus de Markov, les réseaux bayésiens, la logique floue, les algorithmes génétiques, les réseaux de neurones, ou encore les modèles d'intelligence en essaim [22].

Règles et signatures des attaques Plutôt que d'attendre que des anomalies de fonctionnement, de performances, soient détectées, d'autres IDS prennent les devants et mettent en place un système de surveillance du voisinage en établissant un ensemble de règles à ne pas transgresser. Ces règles sont construites à partir de la « signature » des attaques connues c'est-à-dire, pour une attaque donnée, à partir du comportement type adopté par un nœud qui tenterait de mener cette attaque [22].

Les règles peuvent être nombreuses et variées, en voici quelques exemples représentatifs.

- **Règle sur la fréquence des envois** : le délai entre deux paquets consécutifs, ou plus généralement la fréquence des messages émis sur un intervalle de temps par le nœud surveillé, ne doivent pas dépasser un certain seuil (minimal si l'on parle de délai, maximal pour une fréquence). Cette règle empêche un capteur de procéder à des attaques par déluge de paquets, ou bien d'accaparer le médium de transmission, sans être détecté.

- **Règle de retransmission** : les messages reçus par un nœud relai, mais à destination d'un autre nœud, doivent impérativement être retransmis par ce relai au nœud suivant sur la route définie pour ces paquets, ceci afin de détecter les attaques de type *blackhole* ou *sinkhole*.

- **Règle sur le délai de retransmission** : la retransmission en question par le relai ne doit pas être effectuée après un certain délai, afin d'éviter des ralentissements importants dans le réseau (cette règle est très proche de la précédente, qui est bien obligée de se baser sur un délai limite pour considérer que le paquet ne sera pas retransmis).

- **Règle sur l'intégrité** : Le contenu du message retransmis par le relai ne doit pas être altéré. Cette règle est bien entendu inapplicable lorsque le relai est censé effectuer un traitement (agrégation, traitement sémantique...) sur ce contenu avant de le retransmettre. Les attaques visées sont principalement celles visant l'intégrité des paquets.

- **Règle sur la répétition** : un message ne peut être transmis par un nœud donné au plus qu'un certain nombre de fois. Si ce seuil est dépassé, le nœud peut être suspecté de vouloir influencer les résultats collectés par la station de base. Si la répétition est espacée dans le temps, il peut aussi s'agir d'une tentative d'attaque par rejeu de paquets.

- **Règle sur la portée de transmission** : les paquets « entendus » par les nœuds de surveillance ne devraient provenir que de nœuds voisins dans le réseau, et non de nœuds que l'on sait éloignés. Réciproquement, les nœuds voisins ne sont pas censés émettre avec une puissance dépassant un certain seuil. Cette règle peut permettre de détecter des nœuds qui tenteraient de produire une fuite de données vers l'extérieur du réseau, ou bien de mener des attaques dans des zones plus éloignées du réseau (notamment des attaques sur le routage, en prétendant être voisin de nœuds distants, ou bien des attaques par trou de ver).

- **Règle sur les collisions** : pour lutter contre les tentatives de brouillage, un seuil peut être instauré pour le nombre maximum d'échecs de transmission dûs à des collisions. Si ces collisions sont trop fréquentes et que le capteur qui en est à l'origine peut être identifié, il est considéré comme compromis.

- **Règle sur les accès** : si le nombre de tentatives d'un nœud pour s'enregistrer et obtenir un accès à certaines ressources dépasse un certain seuil, une alarme est déclenchée. Cette règle peut notamment s'appliquer à la détection des intrusions au niveau machine, lorsqu'une entité tente et échoue plusieurs fois de se connecter sur le système.

Lorsque les règles sont transgressées, une « alarme » est déclenchée. Suivant le modèle de l'IDS, cette alarme peut n'être qu'une notification interne au capteur qui héberge le système, ou bien il peut partager l'information avec son entourage ainsi que, généralement, la station de base. Les mesures à prendre en guise de réactions seront traitées plus bas.

Les nœuds qui observent le comportement de leurs voisins et veillent à l'application des règles choisies pour le réseau sont désignés de plusieurs façons : on parle de nœuds de contrôle, de nœuds de surveillance, éventuellement de sentinelles ; en anglais, ce seront le plus souvent des *monitoring nodes* (nœuds moniteurs). L'expression *watch dogs*, pour « chiens de garde », est aussi régulièrement employée [105]. Dans nos travaux, les nœuds chargés de cette surveillance sont appelés *cNodes*.

La détection basée sur la signature des attaques est relativement simple à appréhender et à mettre en œuvre. Elle est très efficace pour détecter les attaques connues, dont la signature est transcrite sous forme de règle servant à proscrire les comportements malicieux. Ces règles sont faites de telle façon que le taux de faux positifs soit peu élevé. De plus, cette méthode de détection est centrée sur l'observation du comportement individuel des nœuds voisins des sentinelles, et permet donc le plus souvent de localiser efficacement l'origine d'une attaque. En revanche, elle souffre d'une mauvaise adaptation aux attaques nouvelles, dont le profil n'est pas connu, et risque ainsi de ne pas détecter d'intrusion alors même que les performances du réseau se retrouvent fortement dégradées.

Respect des spécifications Une troisième classe de méthodes de détection regroupe les méthodes, moins fréquentes, basées sur le respect de spécifications réalisées en amont du déploiement du réseau. La détection basée sur les anomalies détecte les écarts de performance, celle basée sur les signatures relève les comportements suspects ; cette troisième catégorie cherche à combiner les avantages de chacune, en se reposant sur des spécifications élaborées par l'opérateur ainsi que sur des contraintes qui caractérisent le bon fonctionnement du réseau. Les écarts au profil normal spécifié permettent de détecter les attaques en identifiant leur effet, tandis que les contraintes fixées par ces spécifications entrainent la détection de comportements individuels malicieux [22].

Comme les spécifications sont réalisées par un opérateur humain (il ne s'agit pas d'un modèle généré par un algorithme), le taux de faux positifs est relativement faible. En revanche, l'élaboration de ces spécifications peut s'avérer très complexe, et prendre beaucoup de temps. Ces méthodes sont relativement peu implémentées dans la littérature.

2.4.3 ▶ Des architectures adaptées

Nous indiquons ici quelques exemples de propositions concrètes de systèmes de détection d'intrusion, sans chercher à couvrir l'ensemble des nombreuses catégories présentées. La plupart d'entre elles se basent sur l'emploi d'architectures (clusterisation, architectures de sécurité) ou de protocoles (routage) déjà existants, auxquelles elles rajoutent des systèmes dédiés à la détection des attaques.

Quelques IDS pour réseaux clusterisés Le système SecCBSN (*Secure Communications of Cluster-Based Sensor Network*, « communications sécurisées des réseaux de capteurs clusterisés ») introduit une architecture de sécurité à laquelle est adjointe un

système de détection d'intrusion pour sécuriser au mieux possible les réseaux clusterisés [59]. Chaque nœud implémente trois modules : l'un intervient pour l'élection du cluster head, réalisée sur le modèle de LEACH. Le deuxième module est chargé d'apporter authentification et chiffrement pour les communications entre les capteurs au sein du réseau, et utilise notamment une version retouchée du certificat TESLA. Ajout notable par rapport aux architectures que nous avions vues jusque là, un troisième module est chargé de la détection des nœuds compromis. Lorsqu'il détecte des comportements anormaux de la part de nœuds donnés, ce module fait propager une alarme jusqu'à la station de base pour l'informer de l'état malicieux de ces capteurs. Ce module se base sur un ensemble assez complexe de règles permettant de vérifier à la fois le bon comportement (non compromission) et le bon fonctionnement (disponibilité) des nœuds voisins. Le système dans son ensemble est finalement proche des mécanismes que nous utilisons, sauf que tous les capteurs réalisent une activité de surveillance.

Notre système en question repose sur les travaux de Lai et Chen, qui ont proposé en 2008 une solution proche mais dont seuls certains capteurs effectuent des opérations de surveillance [77]. Ces nœuds de contrôle, appelés *gNodes* (pour *guarding nodes*) par leurs auteurs, sont sélectionnés au sein du cluster pour effectuer les opérations de surveillance, permettant ainsi aux autres nœuds d'économiser d'autant en énergie. Lorsque le cluster head reçoit plusieurs rapports de la part de *gNodes* distincts (ceci afin de limiter les cas de faux positifs), le CH en vient à considérer le capteur dénoncé comme étant corrompu et prend des mesures en conséquence. Bien que les simulations menées fournissent de bons résultats quant à la préservation des ressources, l'étude ne se penche pas sur le processus utilisé pour la désignation des nœuds de contrôle, et ne traite pas de leur renouvellement dans le temps, ce qui fait que les même capteurs supportent la tâche de surveillance sur de longues durées.

C'est sur ces points que se concentrent la majeure partie des travaux présentés dans cet ouvrage : les *cNodes* que nous utilisons ne sont qu'une version renommée des *gNodes*, pour lesquels a été proposé dans un premier temps un renouvellement périodique [48]. Nous nous attachons, dans le choix des nœuds de surveillance, à répartir au mieux la consommation énergétique dans le cluster ; d'autres études cherchent à désigner ces nœuds de manière à obtenir la meilleure couverture possible, et à maximiser les chances de détection des nœuds compromis à l'aide d'heuristiques pour le placement de l'IDS [66].

Lutter contre les pertes de paquets À travers l'usage plus ou moins direct de systèmes de détection d'intrusion, plusieurs architectures déjà établies ont été adaptées dans le but de faciliter la détection de nœuds corrompus au sein du réseau.

OLSR (*Optimized Link State Routing protocol*) est l'un des protocoles de routage les plus souvent mis en place dans les réseaux de capteurs. Une nouvelle version de ce protocole, appelée DLSR (*DDoS-preventing OLSR*) [90], a été proposée en vue de reprendre ses fonctionnalités tout en ajoutant des mécanismes de détection contre les attaques par déni de service distribuées. Lorsque la station de base reçoit un trafic trop important, qui menace de saturer sa capacité de traitement ou bien qui crée de trop fortes congestions sur ses liens, elle envoie un signal dans tout le réseau en vue de prévenir les nœuds ; le trafic est alors analysé, par chaque nœud relai, à l'aide d'un automate d'apprentissage afin de détecter et supprimer les paquets malveillants. DLSR fait signaler par la station de base la fin de l'attaque, afin que les nœuds cessent les opérations nécessaires au filtrage des paquets.

Basé sur des mécanismes simples, REWARD reprend la règle de retransmission pour assurer qu'aucun nœud ne commet d'attaque de type *blackhole* [70]. À chaque émission de paquet, le nœud émetteur (qu'il soit émetteur d'origine ou relai, sauf s'il est le dernier relai de la route empruntée) vérifie que le destinataire retransmet le paquet sur le saut suivant. Cette procédure ne permet pas de détecter deux attaquants « alignés » sur la route des paquets, qui agiraient de concert, le premier transmettant le paquet au second, sans lever d'alerte lorsque celui-ci se contenterait de supprimer le paquet. Aussi lorsque l'on suspecte que des attaquants coopèrent, REWARD ne se contente plus de ne faire surveiller que ses voisins directs par un capteur, mais propose d'augmenter la puissance de transmission lors du transit du paquet, de sorte que la règle puisse être vérifiée par chaque nœud sur les deux sauts suivant son émission.

Une conséquence des attaques de type *sinkhole*, outre la perte des paquets, est la consommation supplémentaire en énergie que l'attaquant impose à ses voisins. Une étude se base sur ce constat pour établir une méthode « géostatistique » de détection pour ces attaques [108]. Le principe consiste à échantillonner périodiquement des zones du réseau, en demandant aux nœuds leur énergie résiduelle. Un traitement sur ces données permet de déterminer les zones ou l'énergie des nœuds est plus faible qu'elle ne le devrait, et donc où une attaque de type *sinkhole* est peut-être en cours. Ce traitement consiste en l'application d'un modèle de survie, plus précisément une régression de Cox (modèle à risque proportionnel), qui est réalisée soit par la station de base, soit, dans une version adaptée, par des nœuds de confiance au sein du réseau.

Sécurité système Outre les systèmes de détection d'intrusion machine, qui opèrent sur le système d'exploitation du nœud même sur lequel ils s'exécutent, il est possible de faire vérifier l'intégrité des systèmes par d'autres entités. Il existe ainsi une proposition concernant l'usage d'agents mobiles dans le réseau, qui se déplacent de nœud en nœud pour vérifier que le code binaire de leur système n'a pas été altéré [52]. Si des capteurs corrompus sont détectés et que leur état ne peut être immédiatement restauré (soit que l'opération soit impossible, soit que l'agent mobile reçoive l'alarme d'un autre capteur et n'ait pas encore eu le temps d'accéder au nœud affecté), ces agents permettent également de propager l'alerte dans le réseau.

Plusieurs études se proposent de résumer les avancées des solutions proposées en matière de systèmes de détection d'intrusion [87, 86], et fournissent de nombreux autres exemples de ces systèmes. Les travaux de synthèse récents de Butun, Morgera et Sankar, qui classent et détaillent les IDS pour MANET et réseaux de capteurs sans fil [22] sont notamment très complets. Et parce qu'aucun système IDS n'est parfait, on trouve même des travaux réalisés sur les limites de certaines méthodes de détection et des mécanismes de confiance (que nous verrons plus bas), assorties de mesures permettant d'améliorer l'efficacité de ces méthodes [28].

2.5 Réagir aux attaques

Savoir qu'une attaque est en cours dans le réseau est une information de grande valeur ; mais encore faut-il savoir l'utiliser pour réagir efficacement.

2.5.1 ▸ Alertes et exclusion du coupable

Exclure l'attaquant Lorsque l'auteur d'une attaque a été clairement identifié, la réponse la plus souvent employée consiste à l'exclure virtuellement du réseau, c'est-à-

dire à ne plus lui envoyer de paquets, et à ignorer totalement les trames provenant de ce nœud [77]. Cette exclusion peut permettre d'empêcher le nœud corrompu de procéder à ses attaques, du moment que ses paquets peuvent effectivement être ignorés. Pour tout ce qui concerne les comportements cupides, la falsification d'accusés de réception, puis de manière générale l'ensemble des attaques portant sur les couches réseau ou supérieures de la pile TCP/IP, ignorer les paquets émis par le capteur s'avère efficace : puisque les informations qu'il envoie ne sont plus traitées, il n'y a plus de danger. Elle sera sans effet en revanche sur des attaques de type brouillage, ou de création de collisions : non seulement l'attaquant est difficile à identifier, car il ne renseigne pas son identité dans les trames malicieuses, mais en plus le fait de l'« exclure » ne permet en aucun cas de supprimer les collisions produites sur des paquets légitimes par ses émissions.

Identifier l'attaquant Suivant le modèle utilisé, et notamment lorsque la détection est basée sur le relevé d'anomalies, détecter une attaque ne revient pas nécessairement à identifier son auteur. Pour pouvoir exclure l'attaquant, il est pourtant nécessaire de connaitre son identité : la détection d'une anomalie peut donc parfois être suivie de l'activation d'un processus de surveillance des nœuds voisins, afin de repérer quelle entité est à l'origine de l'attaque [22].

Diffuser un message d'alerte Si le système de détection d'intrusion mis en place fonctionne de manière indépendante sur chaque capteur, il est possible que les nœuds ne diffusent pas d'information lorsqu'ils identifient un nœud corrompu. Mais dans la majorité des cas, il est utile de le faire savoir au voisinage ainsi qu'à la station de base [22]. Différents niveaux de diffusion de l'alerte sont donc proposés selon les systèmes. Certains ne déclenchent une alerte qu'à destination de la station de base, d'autres préviennent l'ensemble des agents du réseau. Dans les réseaux clusterisés, la constante est de remonter une alerte au cluster head, qui la transmet à la station de base et la rediffuse au sein de son cluster.

2.5.2 ▶ Rétablissement du réseau

Restauration des systèmes compromis Inscrire les nœuds corrompus sur une liste noire n'est pas nécessairement la seule réponse possible : il est parfois possible de rétablir la configuration initiale des capteurs. Une étude propose, pour un réseau dont les capteurs ne seraient pas trop difficiles d'accès, l'usage d'agents mobiles pour récupérer au mieux d'attaques de déni de service « sur les chemins » [79]. Ces attaques consistent à compromettre plusieurs nœuds, qui à leur tour mènent des attaques par déluge de paquets dans le but de créer des congestions et d'épuiser la batterie de tous leurs voisins, allant ainsi jusqu'à créer de grands espaces vides dans le réseau. La solution proposée repose sur l'usage d'entités mobiles qui sont capables de se déplacer dans le réseau pour rétablir le statut initial des capteurs et si possible récupérer le contenu de leur base de données. Mais ce déplacement prend du temps : si moins de 15 % des capteurs sont suspects, il est envisageable de laisser les agents mobiles tourner dans le réseau pour restaurer leur configuration d'origine. Les nœuds suspects ne sont pas exclus, ce qui limite le risque de faux positifs. Lorsque le taux de capteurs suspects est évalué entre 15 % et 60 %, une liste noire est créée pour limiter les dégâts infligés aux nœuds légitimes le temps que les agents mobiles puissent agir. Enfin, lorsque le seuil critique de 60 % est dépassé, les agents mobiles s'appliquent avant tout à restaurer l'image mémoire et logicielle des nœuds jugés les plus essentiels au bon

fonctionnement du réseau sur des capteurs encore en état de marche, et à entretenir ces derniers.

Résilience du réseau Il n'est pas toujours possible d'annuler les effets d'un attaque en excluant un capteur corrompu du réseau, à plus forte raison si l'attaque est menée depuis l'extérieur. Il faut dans ce cas trouver des méthodes alternatives pour composer avec l'attaque en cours.

Ainsi, l'impact du brouillage de certaines fréquences peut être limité, par exemple en changeant la fréquence utilisée pour le canal de transmission. Si l'attaquant parvient à s'adapter, un changement de fréquences très rapide ou « sauts de fréquence », utilisé par des protocoles de couche de liaison de données à étalement de spectre comme CDMA (*Code Division Multiple Access*, qui consiste à étendre le spectre d'émission et à convenir, entre pairs, d'une séquence pseudo-aléatoire guidant les sauts rapides de fréquences), peut permettre d'échapper au collisions. Le sigle FHSS est utilisé pour cette technique et provient de l'anglais *Frequency-Hopping Spread Spectrum*, en français « étalement de spectre par saut de fréquence », à opposer à DSSS, *Direct-sequence spread spectrum*, c'est-à-dire « étalement de spectre à séquence directe » (DSSS est un autre procédé associé à CDMA, mais joue plutôt sur un changement de phase du signal là où FHSS modifie les fréquences employées ; DSSS ne permet donc pas de résister aussi bien aux attaques par brouillage). Des antennes directionnelles peuvent aussi permettre une meilleure qualité de signal, et rendent parfois possibles les communications au travers d'une tentative de brouillage [98]. Enfin, si les nœuds sont mobiles, ils peuvent tenter de sortir de la zone affectée par les émissions de l'attaquant [98].

Lorsque toute une zone du réseau est sous l'effet d'une attaque de brouillage et se retrouve coupée de la station de base, une possibilité intéressante est d'établir un « trou de ver », légitime cette fois-ci, entre un relai de la zone atteinte et un nœud extérieur, qui sert alors de passerelle vers le reste du réseau [23]. Ce tunnel permet de n'équiper qu'un sous-ensemble réduit de capteurs en matériel nécessaire pour contrer les attaques de brouillage radio, et de compter sur eux pour établir la passerelle salvatrice.

En dehors de ce tunnel, les dispositions évoquées jusque là pour échapper au brouillage sont très couteuses pour des réseaux de capteurs, car elles demandent un équipement ou bien une capacité de calcul conséquents. Une technique plus simple pour réagir à du brouillage « intelligent », dans le cas où l'attaquant n'émet que quelques bits de données pour corrompre un paquet, consiste à ajouter des codes de correction d'erreur aux messages [98]. Ces codes demandent un peu plus de calculs que les sommes de contrôle ; mais tandis que celles-ci ne font que déceler les erreurs, les codes correcteurs permettent parfois de les corriger (et d'éviter ainsi la suppression pure et simple du paquet corrompu, qui mènerait à une demande de retransmission des données).

Laissons à présent de côté les tentatives de brouillage pour nous intéresser au routage : les attaques sur les protocoles sont stoppées par l'exclusion de l'attaquant du réseau, mais les dégâts causés ne se réparent pas seuls. Il faut donc relancer les algorithmes afin de rétablir des routes efficaces. Le problème s'accentue lorsque l'attaquant est parvenu à mettre des nœuds hors service en les poussant à vider leur batterie. Des capteurs peuvent également se retrouver coupés du reste du réseau. Il faut alors trouver de nouveaux chemins pour les données, éventuellement en recréer si les nœuds sont mobiles. Un certain niveau de résilience est donc nécessaire dans

la conception des protocoles de routage, de sorte qu'ils soient capables de s'adapter à l'évolution du réseau.

Certains mécanismes proposent même l'intervention d'un opérateur humain pour déclencher une modification de certaines routes et « emprisonner » le trafic frauduleux, qui vise à créer des congestions dans le réseau, sur des nœuds *Shield* (« bouclier » en anglais) qui se contenteront de le supprimer [107]. Si l'intervention d'un opérateur humain n'est pas forcément souhaitable dans les réseaux de capteurs, ce mécanisme a pour avantage de désarmer l'attaque sans mettre au courant l'attaquant de son échec (tandis qu'une exclusion directe du réseau serait plus facile à remarquer).

2.5.3 ▶ Notion de confiance

Il existe, liée aux IDS, une notion intéressante de « confiance » qui peut être mise en place dans un réseau de capteurs. Plutôt que de considérer les capteurs comme « tout blancs » ou « tout noirs », beaucoup de systèmes de détection leur attribuent un score de « réputation », qui évolue dans le temps, et reflète la « confiance » (en anglais : *trust*) que l'on peut avoir en tel nœud.

Cette notion de confiance intervient en plusieurs endroits. Elle fait parfois partie intégrante du système de détection. Prenons ainsi l'exemple d'un réseau utilisant un système basé sur les signatures d'attaques : les nœuds exécutant l'IDS surveillent leurs voisins et, lorsqu'une règle est enfreinte de manière exceptionnelle, ne vont pas immédiatement déclarer le coupable comme étant compromis. À la place, son score de réputation diminue. Ce n'est que lorsque ce score atteint un certain seuil minimal que le nœud est considéré comme étant corrompu. Comme nous l'avons évoqué plus haut, la surveillance peut être réalisée de façon distribuée, ou chacun de son côté. Il en va de même pour la réputation, dont le score de ses voisins peut très bien n'être établi par un nœud que pour son propre usage. Mais elle se prête particulièrement bien à la distribution de la tâche : les nœuds peuvent échanger entre eux le niveau de confiance qu'ils attribuent à leurs différents voisins, et adapter leurs valeurs en fonction des données reçues : si A se méfie de B et en informe C, tandis que C a pleinement confiance en A, alors C va légèrement décrémenter le score de réputation qu'il attribue à B. Si par ailleurs C constate lui aussi des manquements aux règles commis par B, alors les nœuds A et C peuvent décider de concert que B doit être considéré comme corrompu : la confiance est ici transitive (dans une certaine mesure).

Mais la confiance n'est pas qu'une modulation de la détection : elle permet aussi de prendre des décisions plus adaptées à la situation du réseau. Ainsi, il est tout à fait possible d'introduire un mécanisme basé sur la confiance pour protéger l'élection des cluster heads dans le réseau [31]. Si les nœuds procèdent à une élection du CH, ils vont alors prendre en compte différents paramètres (distances, réserves énergétiques…) pour déterminer leur vote, parmi lesquels le score de réputation de leurs voisins jouera un rôle prépondérant : ils ne vont pas voter pour un nœud qui est susceptible d'être compromis, mais au contraire pour ceux en lesquels ils ont le plus confiance.

Les mêmes décisions peuvent être appliquées à la création de routes pour la retransmission des paquets. Si un capteur a une mauvaise réputation, c'est-à-dire s'il est peu fiable, par exemple parce qu'il a déjà perdu de nombreux messages, mieux vaut éviter de le choisir comme relai. Certains protocoles de routage intègrent, dès leur déploiement, des mécanismes de confiance [138].

Pour prendre ces décisions de manière efficace, il est nécessaire d'adapter en temps réel le score de réputation de chaque voisin d'un nœud : les mécanismes de confiance

sont donc indissociables de la surveillance propre aux IDS. Ils se prêtent bien à la détection basée sur les signatures d'attaques, puisqu'il suffit de mettre à jour les scores lorsque des infractions sont constatées (ou de les remonter au bout d'une période donnée sans faute observée). Ils peuvent aussi être utilisés en conjonction avec des approches plus formelles utilisées notamment dans les modèles de détection par apprentissage [39, 92] : la logique floue, les réseaux bayésiens, ou même des éléments de la théorie des jeux sont utilisés dans de nombreux systèmes pour établir des scores de confiance efficaces.

2.6 Une vision d'ensemble

La prévention et le couple détection / réaction sont les deux axes de lutte contre les attaques par déni de service, et viennent clore cette section. Ces mesures doivent être menées conjointement avec des mécanismes de protection des autres propriétés : l'authentification, mais aussi l'intégrité des messages échangés par exemple, doivent être assurées pour parer à plusieurs types d'attaques. Le chapitre 3 se termine également ici : après avoir présenté ces différentes problématiques de sécurité, puis en particulier les attaques et les principales contre-mesures concernant le déni de service, c'est un aperçu global de la sécurité dans les réseaux de capteurs qui a été établi. Il vient se joindre aux éléments du chapitre 2, qui introduit le fonctionnement générique des capteurs et les problèmes qu'ils apportent.

En résumé, ce sont les points suivants qui ont été présentés dans ces deux chapitres.

- Les réseaux de capteurs sans fil comportent des capteurs de petite taille, faibles en capacité de calcul, en mémoire et en batterie, qui communiquent entre eux par voie hertzienne.
- Les limites en ressources de ces capteurs, par ailleurs utilisés dans de nombreuses applications, imposent dès le déploiement du réseau d'utiliser des algorithmes et des protocoles adaptés à leurs capacités.
- La partition en clusters est une opération couramment utilisée pour faciliter et rendre plus efficace la gestion de ces réseaux.
- Ces réseaux sont par ailleurs soumis à des problématiques de sureté, de résilience, et surtout de sécurité : confidentialité, authentification, intégrité, protection contre le rejeu et disponibilité des communications et des services en sont les principaux exemples.
- Il existe de nombreuses solutions pour garantir ces propriétés, toutes avec leur avantages et leurs inconvénients. La sécurité, en informatique, est toujours un compromis entre les performances attendues et le cout supplémentaire impliqué par les mécanismes mis en place.

Appliquées aux réseaux de capteurs, ces solutions se doivent naturellement d'être économes en ressources, et particulièrement en énergie nécessaire. Armés de cette vision d'ensemble des capteurs, des contraintes en ressources et des exigences soulevées par les questions de sécurité, nous sommes maintenant à même d'introduire de nouveaux mécanismes pour préserver encore davantage la batterie des capteurs.

4

Sélection aléatoire des *cNodes*

Les réseaux de capteurs, leur partition en clusters, et les différentes solutions existantes pour lutter contre le déni de service forment une charpente sur laquelle il est possible de construire encore. Les chapitres qui viennent, à commencer par celui-ci, présentent les différentes contributions apportées au cours de la thèse. Ces contributions sont principalement centrées sur le renouvellement et l'établissement d'un processus efficace de sélection pour les nœuds de surveillance d'un réseau de capteurs.

La première d'entre elles détaille l'architecture ainsi que les mécanismes employés, met en place le renouvellement de la sélection, et propose un processus aléatoire pour la désignation des nœuds de surveillance. Elle est évaluée par des simulations à évènements discrets, puis à l'aide d'outils plus formels, le tout organisé de la façon suivante :

- dans un premier temps est présenté un jeu de résultats numériques obtenus par des simulations réalisées avec le logiciel ns, avec et sans renouvellement des nœuds de surveillance, afin d'éprouver la méthode proposée ;
- dans un second temps intervient la modélisation. Le système est d'abors l'objet d'une première représentation sous forme de chaines de Markov : des mesures analytiques sont obtenues à partir des états stationnaires, pour obtenir la probabilité de la détection des attaques dans le cadre de ce modèle ;
- face aux limites de ce premier outil formel, une seconde représentation du mécanisme de détection est exprimée à l'aide de réseaux de Petri stochastiques généralisés, et accompagnée de propriétés établies à l'aide de logique stochastique avec automates hybrides.

1 Mise en place de la détection des attaques

1.1 Hypothèses de travail

Sauf indication contraire, les travaux exposés dans le présent chapitre, ainsi par ailleurs que dans les chapitres 5 et 6, reposent sur les hypothèses suivantes :

- Le contexte est celui d'un réseau de capteurs sans fil constitué de nœuds possédant tous les mêmes caractéristiques physiques et techniques et limités en ressources disponibles, ainsi que d'une station de base « non limitée » en ressources.
- Pour en faciliter la gestion, ce réseau est partitionné en clusters. Les mécanismes déployés au sein de chaque cluster sont identiques, si bien qu'il est possible, pour les décrire, de ne traiter que le cas d'un seul cluster.
- Au sein des clusters toujours, tous les capteurs peuvent atteindre le cluster head en un seul saut : ils n'ont donc pas besoin de moduler la puissance de leurs transmissions. Les cluster heads sont les seuls à recourir à cette possibilité pour atteindre directement la station de base, si aucun protocole de routage inter-clusters n'a été mis en place.
- La mobilité des nœuds est faible, voir nulle ; elle est en tout cas négligeable par rapport à l'échelle temporelle utilisée dans les protocoles mis en place, et les nœuds sont donc considérés comme statiques.
- Le but final recherché est la détection d'un attaquant qui mènerait un assaut contre le réseau en introduisant ou en compromettant un capteur (il s'agit donc d'un attaquant interne, équipé d'un matériel de caractéristiques identiques aux autres capteurs).

1.2 Objectifs à atteindre

Le but de la solution proposée est de fournir un moyen efficace de détecter un nœud compromis dans le réseau, qui tenterait par exemple de saturer les capacités de communication du réseau, en envoyant un flux de données plus important que les nœuds « honnêtes ». Plusieurs attaques peuvent être détectées par l'application de règles sur le trafic qui transite au sein du réseau (voir le paragraphe correspondant au chapitre 3, sous-section 2.4.2). Le modèle qui vient d'être donné sera celui repris à titre d'exemple tout au long de la description de la méthode.

Dans ce contexte, le modèle de l'attaque étudié est donc le suivant : un capteur compromis, sous la direction de l'attaquant, transmet un volume de données (que ce soit par la taille ou bien par le nombre de paquets envoyés) bien plus élevé que ce qu'il devrait, ce qui peut mener :

- à l'accaparement du canal de transmission, ce qui peut empêcher les nœuds légitimes d'envoyer leurs propres messages ;
- à des congestions sur le trafic, par exemple si les paquets sont envoyés à un rythme trop soutenu pour que le CH ait le temps de tous les traiter ;

- à l'épuisement des nœuds en écoute, et notamment du cluster head submergé de données à retransmettre ;
- à l'altération de la représentativité des données obtenues par l'exploitant, surtout dans le cas où les paquets contiennent des données erronées.

L'efficacité d'une méthode de détection de ce nœud compromis peut être mesurée sur deux aspects :

- le taux de détection du ou des nœud(s) compromis dans le réseau ;
- la durée de vie du réseau.

Le second critère peut lui-même être exprimé de plusieurs façons. Par exemple, il peut être directement associé à la durée de vie du dernier capteur restant dans le réseau ; mais ce capteur seul n'est plus forcément à même de fournir un service efficace. Aussi la durée de vie du réseau est-elle souvent définie comme la durée écoulée entre le déploiement du réseau et l'instant où le premier capteur à manquer d'énergie s'éteint. C'est cette deuxième définition qui sera utilisée par la suite. Elle implique que pour obtenir une grande durée de vie, la consommation en énergie du réseau doit être équitablement répartie entre un aussi grand nombre de nœuds que possible.

Pour obtenir une méthode efficace sur les deux points évoqués, nous proposons dès lors d'accompagner la mise en place des nœuds de surveillance par deux mécanismes : une partition hiérarchique récursive des nœuds du réseau d'une part, et une sélection dynamique des « sentinelles » d'autre part.

1.3 Partition hiérarchique du réseau

Le fait de clusteriser le réseau de capteurs tel que présenté au chapitre 2, sous-section 2.2 permet de limiter la consommation en énergie de la plupart des nœuds du réseau, puisque seuls les cluster heads se retrouvent à effectuer des émissions sur des distances plus longues que le rayon d'un cluster. La partition récursive permet d'obtenir une gestion encore plus efficace des clusters, puisqu'elle introduit plus de granularité. Nous proposons, pour notre solution (et pour nos tests, voir section 2), une hiérarchie à deux degrés : nous avons clusterisé notre réseau en utilisant k-LEACH, en fixant $k = 2$ (voir figure 4.1). Ceci permet, à notre sens :

- d'économiser davantage d'énergie par rapport à l'usage de LEACH dans sa version simple ;
- d'obtenir, en choisissant des nœuds de surveillance dans chacun des 2-clusters, une meilleure couverture du réseau par les sentinelles, et de maximiser ainsi la probabilité de détection des nœuds compromis dans le réseau.

En pratique, k doit être adapté en fonction du nombre de capteurs dans le réseau, de sa superficie, de la puissance et de la consommation énergétique des nœuds, et même des applications utilisées dans le réseau. Des expériences ayant recours au matériel considéré sont indispensables pour déterminer les valeurs numériques optimales pour un réseau donné, ce qui empêche de fournir ici une estimation numérique : nous nous en tenons donc au degré 2.

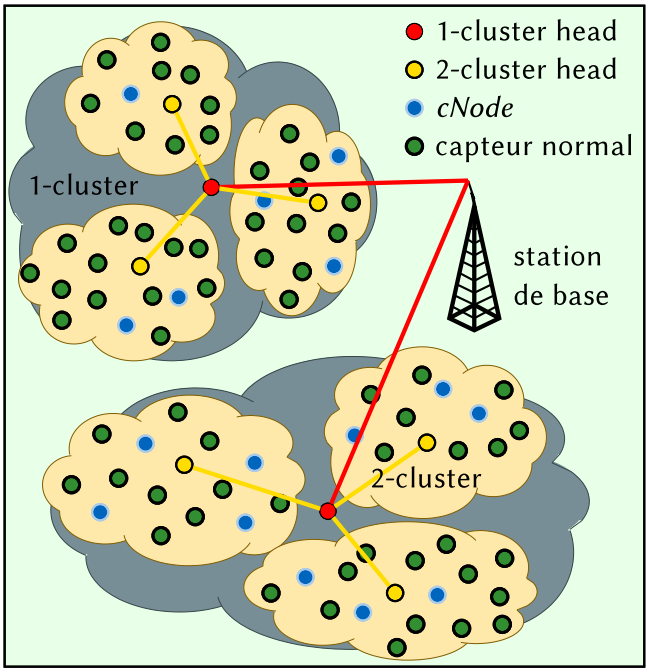

FIG. 4.1 : Schéma du réseau avec deux niveaux de partition

1.4 Des *cNodes* pour surveiller le trafic

La partition hiérarchique d'un réseau en clusters établit déjà une distinction entre deux rôles que peuvent assumer les capteurs : celui de nœud « normal », qui poursuit ses opérations de captage et de retransmission des données de façon classique, et celui de cluster head, qui se retrouve chargé d'assurer la liaison entre les membres de son cluster et le restant du réseau. À côté des nœuds « normaux » et des cluster heads, un troisième rôle va être attribué à certains capteurs, afin d'assurer la détection des attaques par déni de service. Certains capteurs se voient ainsi confier le rôle de « nœuds de contrôle », ici nommés *cNodes* (introduits sous le nom de *gNodes* dans [77]), et sont chargés de surveiller le trafic entrant et sortant des cluster heads.

Si un *cNode* constate qu'un capteur enfreint les règles fixées, par exemple en envoyant, au cours d'une unité de temps, plus de données à leur CH qu'une valeur de seuil déterminée $S_{\text{débit}}$, il retient que le nœud a eu un comportement anormal (il y a eu un écart important à la moyenne de la quantité de données envoyées par unité de temps). Lorsque ce nœud réalise un certains nombres d'écarts, c'est-à-dire lorsque le nombre de comportement anormaux détectés par un *cNode* dépasse une valeur de seuil $S_{\text{écarts}}$, le ou les nœuds de contrôle qui ont repéré ces écarts considèrent alors le capteur comme compromis. Chaque *cNode* ayant détecté un nœud compromis envoie un message d'avertissement à son cluster head, comme représenté sur la figure 4.2. Ici encore, un seuil S_{alertes} est fixé pour le nombre minimum d'alertes qu'un CH doit recevoir avant de considérer un nœud comme malveillant, et ce afin d'éviter qu'un nœud compromis se faisant passer pour un *cNode* ne déclare tous ses voisins comme compromis. Plusieurs *cNodes* distincts doivent donc détecter les écarts d'un nœud pour que celui-ci soit effectivement considéré comme compromis par le cluster head.

Une fois que la malveillance d'un capteur est effectivement actée par un cluster head, la charge incombe à ce dernier de prendre des mesures pour mitiger l'attaque. En pratique, la solution utilisée ici est simple : tous les messages en provenance des nœuds considérés malveillants par le cluster head sont ignorés (le CH se met immédiatement à les ignorer, et il prévient les capteurs du cluster afin qu'ils puissent en faire autant).

Les *cNodes* comparent également la taille des trafics entrant et sortant du cluster head. En cas d'écarts importants (en tenant compte des éventuels facteurs de compression des données et des messages ignorés des nœuds considérés compromis), ils peuvent être amenés à penser que le cluster head est lui-même malveillant. Dans ce cas, l'élection d'un nouveau CH dans le cluster est déclenchée.

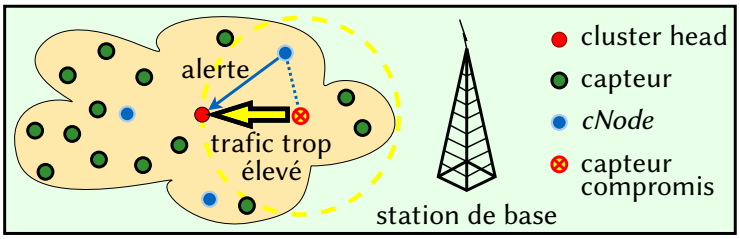

FIG. 4.2 : Schéma : exemple de détection d'un comportement anormal et envoi d'une alerte au CH (un seul cluster représenté)

1.5 Sélection dynamique des *cNodes*

1.5.1 ▶ Motivations

La partition du réseau permet de limiter aux seuls cluster heads les efforts énergétiques dus aux transmissions longues. L'inconvénient est que, justement, ces CH vont épuiser leur batterie beaucoup plus rapidement que s'ils étaient restés des nœuds ordinaires, tirant ainsi le réseau vers une fin de vie précoce. Les nœuds ne sont pas toujours désignés CH « à vie » : ainsi LEACH renouvelle périodiquement la partition du réseau, choisissant à chaque itération de nouveaux cluster heads et entrainant la sélection d'un nouveau lot de *cNodes*.

Mais tous les protocoles de clusterisation ne prévoient pas nécessairement de renouvellement régulier de la partition. De plus, s'ils n'émettent pas sur de longues distances, les *cNodes* se retrouvent eux aussi à consommer plus d'énergie que les nœuds normaux, puisqu'ils doivent sans cesse rester à l'écoute des transmissions qui les entourent, et analyser le débit de chacun de leurs voisins. Pour répartir cette nouvelle charge, qui touche davantage de nœuds, nous proposons d'élire également les *cNodes* de façon dynamique, sur une période plus courte que celle, éventuelle, qui correspond à la clusterisation du réseau.

Il est vrai que le renouvellement périodique des *cNodes* rajoute des contraintes et augmente le volume des messages de contrôle à échanger dans le réseau. Pour certaines applications des capteurs, ceci peut représenter un inconvénient que la répartition de la charge ne vient pas contrebalancer ; mais de manière générale la préservation des réserves énergétiques est primordiale dans les réseaux de capteurs, et un meilleur équilibre de la charge obtenu pour un prix relativement modeste ne semble pas superflu. Autre remarque : en matière de détection d'intrusion, les IDS les plus récents se vantent souvent de n'engendrer qu'un faible surplus de consommation dans

le réseau [84], et la question se pose donc de savoir si la répartition de la charge par un renouvellement dynamique reste pertinente ; nous verrons en section suivante qu'avec un système aussi simple que nos *cNodes*, les simulations annoncent un gain important sur la répartition de la charge. Il faut également garder à l'esprit que les mécanismes de renouvellement périodique et de sélection des nœuds de surveillance ne sont pas dépendants du mécanisme de détection utilisé (c'est-à-dire des *cNodes* qui appliquent des règles sur le trafic analysé). Ainsi, si un IDS « concurrent »

- présente une bonne capacité à économiser l'énergie du réseau,
- s'exécute seulement sur un sous-ensemble des nœuds du réseau,
- et ne nécessite pas, pour fonctionner, d'équipement distinct du matériel de base des capteurs,

alors la dynamique et les méthodes de sélection présentées dans cette thèse peuvent très bien être mises en œuvre pour optimiser détection et consommation.

Un dernier point reste à signaler. Cette solution de renouvellement périodique améliore aussi la sécurité du réseau : si un attaquant cherche à compromettre tous les *cNodes* du cluster, pour pouvoir poursuivre son attaque discrètement, le renouvellement dynamique apporte une parade efficace à l'attaque en complexifiant largement le nombre de cibles à corrompre.

1.5.2 ▶ Le choix par le hasard

Le renouvellement périodique des capteurs de surveillance peut être réalisé de plusieurs façons. Dans ce chapitre, une façon de faire très simple est employée : la sélection des *cNodes*, pour chaque période, est — théoriquement — aléatoire. L'aléatoire est une notion délicate en informatique ; nous nous contenterons plutôt d'un algorithme capable de générer des nombres dits « pseudo-aléatoires ». Une façon de procéder, par exemple, est de recourir à un algorithme capable de générer une suite de nombre d'apparence aléatoire, utilisés pour déterminer quels seront les *cNodes* pour la période à venir [12]. Cet algorithme se doit de respecter les points suivants :

- il doit nécessiter peu de calculs ;
- sa période doit être grande ;
- les valeurs générées doivent être indépendantes et distribuées uniformément.

Nous nous sommes tournés vers un type connu de générateurs, les *Multiplicative Linear-Congruential Generators* (ou MLCG), qui possèdent effectivement une grande période [68]. Plus précisément, le générateur retenu est de type $x_n = a \cdot x_{n-1} \bmod m$, où :

- le nombre pseudo-aléatoire x_n est généré à partir de son prédécesseur x_{n-1} ;
- m est de la forme 2^k, ce qui facilite le calcul de mod m (troncature à droite du résultat sur k bits) ;
- a doit être de la forme $8 \cdot i \pm 3$ (avec i un entier positif) pour que le générateur ait la période maximale, égale à 2^{k-2} ;
- x_0 (la « graine », ou *seed* en anglais) doit être un entier impair (toujours pour avoir la meilleure période).

La méthode de Schrage permet de reformuler le générateur sous une forme qui prévient tout problème de débordement [1] lors de l'implémentation [68]. Le générateur est alors écrit sous la forme suivante : $a \cdot x \bmod m = g(x) + m \cdot h(x)$, avec :

$$\begin{cases} g(x) = a \cdot (x \bmod q) - r \cdot (x \operatorname{div} q) \\ h(x) = (x \operatorname{div} q) - (a \cdot x) \operatorname{div} m \\ q = m \operatorname{div} a \\ r = m \bmod a \end{cases}$$

Voici finalement un exemple de générateur (il s'agit de celui qui a été implémenté pour réaliser les tests présentés en section 2) :

$$x_n = 11 \cdot x_{n-1} \bmod 2^{16}$$

Sa période est égale à 2^{14}. Ce générateur peut alors être utilisé pour calculer des nombres pseudo-aléatoires allant de 0 à 65535 qui, ramenés sur le nombre attendu de *cNodes* dans le cluster ou bien comparés à un seuil de probabilité, permettent de déterminer quels vont être les nœuds élus pour la période en cours. Il convient naturellement d'attribuer une graine différente à chaque nœud du réseau, afin d'éviter que tous les nœuds produisent la même suite de nombres pseudo-aléatoires. Ceci peut être réalisé en initialisant l'algorithme avec une mesure physique — après tout, il s'agit de capteurs ! — dont la valeur est ramenée à l'entier impair le plus proche.

Mais une question subsiste : quelle va-t-être l'entité chargée d'exécuter cet algorithme et de procéder à l'élection des *cNodes* ? Trois méthodes différentes sont ici proposées : une auto-élection distribuée, une élection centralisée au niveau des CH, ou alors une élection centralisée au niveau de la station de base.

1.6 Processus de sélection des *cNodes*

1.6.1 ▶ Auto-élection distribuée

Une première solution consiste à réutiliser l'algorithme d'auto-désignation mis en œuvre par LEACH, comme présenté sur la figure 4.3. Chaque nœud non CH choisit un nombre pseudo-aléatoire compris entre 0 et 1. Si ce nombre est en dessous du pourcentage de *cNodes* désirés dans le réseau (et fixé au moment de l'implémentation par l'utilisateur), ce nœud devient un *cNode*. Autrement, il reste un capteur normal.

Cette méthode présente deux inconvénients. Tout d'abord, elle impose à chaque nœud de calculer un nombre pseudo-aléatoire — calcul parfois couteux —, ce qui n'est pas nécessaire avec les deux autres méthodes. Ensuite, chaque nœud choisit lui-même de se désigner (ou non) *cNode*, sans tenir compte de la décision de ses voisins. Si bien que l'élection des *cNodes* ne prend en compte à aucun moment la clusterisation du réseau. Il est très peu probable que les *cNodes* ainsi élus soient uniformément répartis entre les différents 2-clusters du réseau. Il est même possible que certains 2-clusters se retrouvent totalement dépourvus de *cNodes* (et se retrouvent ainsi dans l'incapacité de détecter une attaque).

Ce second point peut être contourné en menant une élection en deux étapes. Dans un premier temps les nœuds choisissent d'endosser, ou non, le rôle de *cNode* ; les

[1] Un débordement désigne le dépassement, au cours des calculs, de la valeur d'entier maximale gérée par le système, et se traduit par des erreurs de calcul.

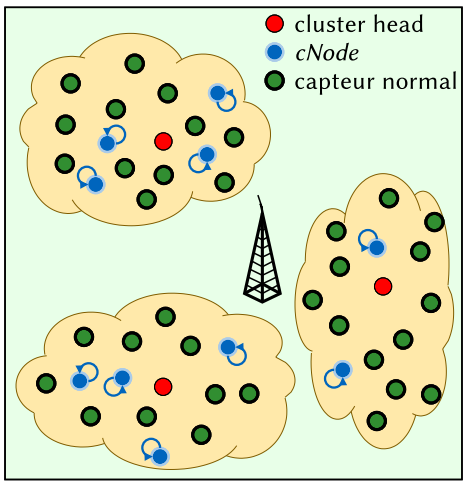

FIG. 4.3 : Auto-élection des *cNodes*

cNodes élus signalent alors leur statut au 2-CH auquel ils sont associés. Au cours de la seconde étape, les 2-CH nomment des *cNodes* supplémentaires dans leurs 2-clusters respectifs si le nombre de signalements reçus au regard du nombre de nœuds dans les 2-clusters est en dessous d'un pourcentage minimal.

1.6.2 ▶ Élection centralisée au niveau des cluster heads

La seconde méthode proposée consiste à décharger les nœuds classiques du processus d'auto-désignation, pour le remplacer par une nomination autoritaire réalisée par les 2-cluster heads (voir figure 4.4). Chaque CH désigne alors un nombre de *cNodes* correspondant au pourcentage indiqué par l'utilisateur. Par exemple, si l'on souhaite obtenir dix pour cent de *cNodes*, et qu'un 2-cluster comporte cent capteurs, le CH de ce cluster va produire dix nombres pseudo-aléatoires distincts qui seront associés aux

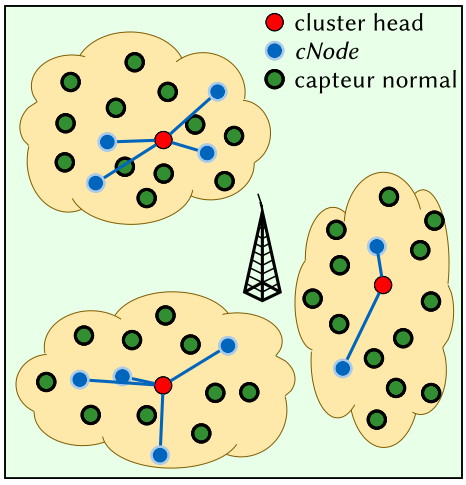

FIG. 4.4 : Élection des *cNodes* réalisée par les cluster heads

identifiants de dix nœuds du cluster. Le CH envoie alors un message à chacun de ces dix nœuds pour leur ordonner de remplir le rôle de *cNode* sur la période en cours.

Cette méthode est plus efficace en ce qui concerne les calculs puisque seuls les 2-CH ont à appliquer un algorithme de calcul de nombres pseudo-aléatoires. Cependant, elle présente un autre désavantage : si l'un de ces CH est compromis par un attaquant, il ne déclarera sans doute aucun *cNode* susceptible de le détecter, laissant ainsi tout son cluster à la merci d'autres attaques. Comme l'algorithme LEACH désigne à tour de rôle chaque capteur, sur un cycle complet, pour accomplir le rôle de CH, tout nœud compromis deviendra logiquement CH à un moment ou l'autre. Le problème du CH ne désignant aucun *cNode* peut donc légitimement se poser pour n'importe quel nœud compromis du réseau. Qui plus est, il faut garder à l'esprit que rien n'empêche un nœud compromis de se déclarer cluster head à chaque ronde de l'algorithme LEACH.

Cette méthode est néanmoins celle qui est implémentée dans les travaux de la référence [48], et que nous reprenons pour nos tests (dont les résultats sont disponibles en section 2).

1.6.3 ▶ Élection centralisée au niveau de la station de base

L'élection centralisée peut également être effectuée au niveau de la station de base, comme représenté sur la figure 4.5. Les 2-CH du réseau font alors remonter à la station de base la liste des nœuds de leurs 2-clusters respectifs, et celle-ci retourne une liste de *cNodes* pour chacun d'entre eux. Les capacités (en mémoire, calcul, énergie) de la station de base étant considérées comme illimitées, cette méthode a pour avantage de déporter toutes les tâches couteuses dans un environnement « sans contraintes ». Le calcul des nombres pseudo-aléatoires par la station de base ne lui coute, pour ainsi dire, rien.

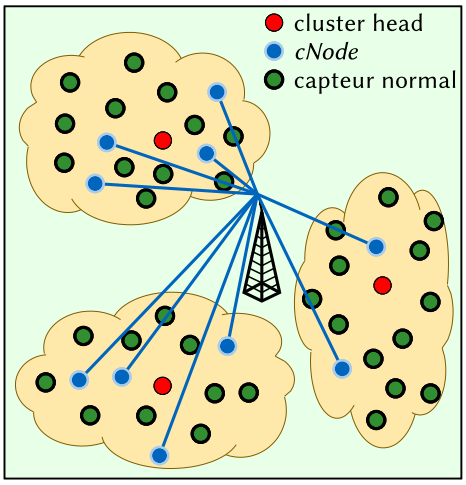

Fig. 4.5 : Élection des *cNodes* réalisée par la station de base

En revanche cette méthode n'offre pas une meilleure fiabilité que la précédente. Si un nœud compromis se déclare en temps que CH, il risque fortement d'annoncer à la station de base que son cluster est vide (il enverrait une liste vide en lieu et place de la liste des capteurs qui sont réellement présents dans son cluster). Dans ce cas, la station de base ne déclarera aucun *cNode* dans le cluster attaqué, et le CH compromis

ne sera pas détecté. Pour parer à cette éventualité, la station de base pourrait agir autrement lorsqu'elle reçoit une liste de capteurs vide. Plus spécifiquement, elle devrait considérer que les nœuds dont elle n'a pas eu de nouvelles *via* les listes des CH ne sont peut-être pas simplement morts, mais escamotés par un cluster head corrompu. Ces nœuds disparus devraient donc être considérés comme potentiellement éligibles, de sorte que certains d'entre eux au moins soient déclarés *cNodes*.

L'inconvénient majeur de cette méthode est le fait que la nature distribuée de l'élection (avec tous les avantages qu'apporte un algorithme distribué) est complètement perdue.

Les avantages et inconvénients de chacune de ces trois méthodes présentées sont résumés dans la table 4.1.

Tab. 4.1 : Avantages et inconvénients de chaque méthode

Méthode	Avantages	Inconvénients
Auto-élection des *cNodes* (selon le modèle de LEACH)	• Très simple à mettre en œuvre • Pas d'envoi de paquets à travers le réseau pour désigner les *cNodes*	• Chaque nœud doit calculer un nombre pseudo-aléatoire • Ne tient pas compte de la topologie du réseau
Élection des *cNodes* par les cluster heads	• Seuls les CH calculent les nombres pseudo-aléatoires	• Si le CH est compromis, il ne désigne aucun *cNode* (un nœud compromis peut se déclarer CH à chaque ronde de LEACH)
Élection des *cNodes* par la station de base	• Aucun calcul réalisé par les nœuds • Distribution spatiale idéale des *cNodes*	• Si le CH est compromis, il déclare un cluster vide • Perte de l'aspect décentralisé de l'algorithme

2 Résultats numériques

2.1 Modèle de la simulation

Afin d'étudier les avantages concrets qu'apporte l'élection dynamique des *cNodes*, nous avons réalisé une série de simulations à l'aide du logiciel ns [123, 124]. Les résultats obtenus sont présentés dans cette section. Ils sont issus de tests simulant le comportement d'un cluster qui se présente sous la forme d'une grille régulière de dix nœuds de côté par dix (voir figure 4.6). Nous supposons, lorsque la question se pose, que c'est le protocole LEACH qui a été mis en œuvre pour établir ce cluster.

Les caractéristiques du cluster étudié sont les suivantes :

- la grille est un carré de côté de longueur a ;
- le cluster head est placé au centre géométrique de la grille ;
- la grille contient cent capteurs, disposés régulièrement à chaque nœud de la grille ;

- chaque capteur peut communiquer directement avec le cluster head ; autrement dit, chaque nœud a une portée au moins équivalente à la distance $\frac{a\sqrt{2}}{2}$, soit une sphère de rayon égal à la moitié de la diagonale du carré. Chaque capteur, y compris ceux situés dans les coins du carré, peut ainsi communiquer avec le cluster head sans intermédiaire. Aucun ajustement de la puissance de transmission n'est effectué par les capteurs ;

- les *cNodes* sont élus selon la méthode statique (unique sélection en début de simulation) ou bien dynamique (élection renouvelée périodiquement), selon le cas.

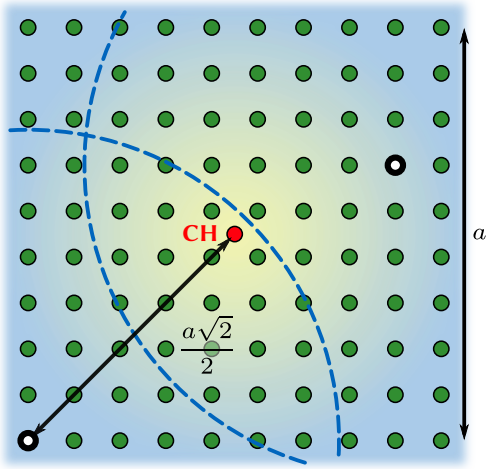

FIG. 4.6 : Cluster sous forme de grille régulière 10×10, de côté de longueur a

2.2 Taux de détection des nœuds compromis

Pour mesurer l'efficacité des *cNodes*, nous avons effectué plusieurs tests avec ns. La table 4.2 indique les valeurs (ou les fourchettes de valeurs) utilisées comme paramètres au cours des différentes simulations.

TAB. 4.2 : Paramètres de simulation

PARAMÈTRE	VALEUR
Durée de la simulation	100–3 600 s
Taux d'émission	10–800 kbits/s
Taille des paquets	500–800 octets
Nombre de nœuds	100 (+ cluster head)
Nombre de *cNodes*	0–30
Nombre de nœuds compromis	1

Pour la première simulation, les nœuds sont élus de façon dynamique sur dix périodes consécutives. Nous avons supposé que la génération de trafic suit une loi de Poisson (la génération est aléatoire, mais on connait sa valeur moyenne). On nomme λ_i le paramètre de cette loi appliquée au nœud i (et λ_c est le nombre moyen de messages envoyés par seconde par le nœud compromis). La figure 4.7 présente le taux de détection du nœud compromis pour différentes quantités de *cNodes* dans le réseau, par période. Au cours de la seconde période de la simulation impliquant dix *cNodes*, nous obervons qu'aucun *cNode* n'a détecté le nœud compromis : les *cNodes* n'étaient donc pas répartis efficacement dans le cluster. Lorsque leur nombre monte à quinze, il y a, en moyenne sur les périodes, trois *cNodes* qui détectent le capteur corrompu. Lorsque ce nombre s'accroit encore, il n'y a plus d'évolution notable. Quinze *cNodes* semblent donc être une mesure idéale pour notre réseau, si l'on considère une élection statique. En revanche, on peut très bien descendre à dix *cNodes* si l'on envisage un processus d'élection dynamique : le nœud compromis ne sera peut-être pas détecté sur une période donnée, mais le renouvellement rapide des *cNodes* permet de s'assurer qu'il sera détecté rapidement sur la période suivante. On note par ailleurs que jusqu'à quarante pour cent (soit quatre nœuds sur dix) ont pu détecter simultanément le nœud compromis sur les périodes 7, 8 et 9.

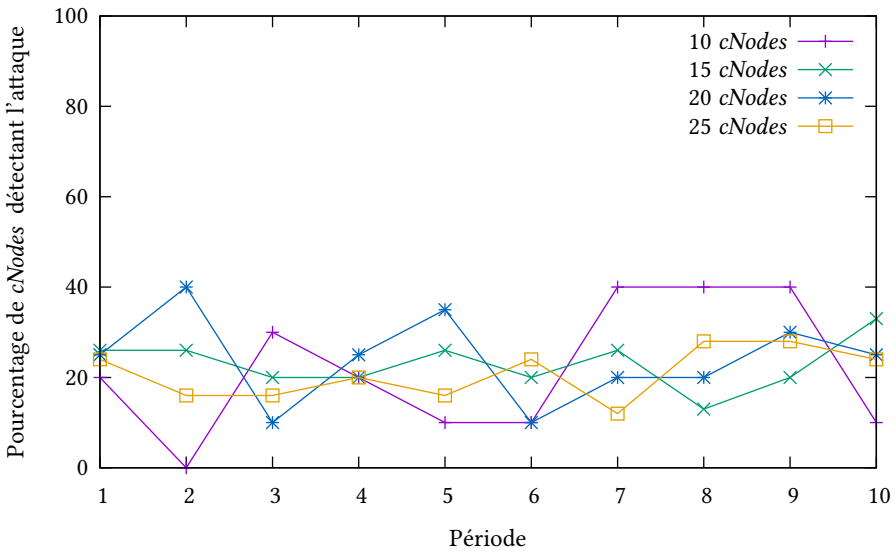

Fig. 4.7 : Taux de détection pour différents nombres de *cNodes*

La valeur de λ_c a elle aussi un rôle à jouer dans la détection : plus le nœud compromis émet, plus il est probable que les *cNodes* mesureront efficacement un dépassement par rapport au seuil théorique d'un nœud honnête. Mais ici encore, le renouvellement rapide des *cNodes* apporte une meilleure détection, puisque statistiquement, davantage de voisins proches du nœud compromis se retrouveront *cNodes* à un moment ou l'autre au cours de la simulation. La courbe présentée en figure 4.8 illustre cet état de fait.

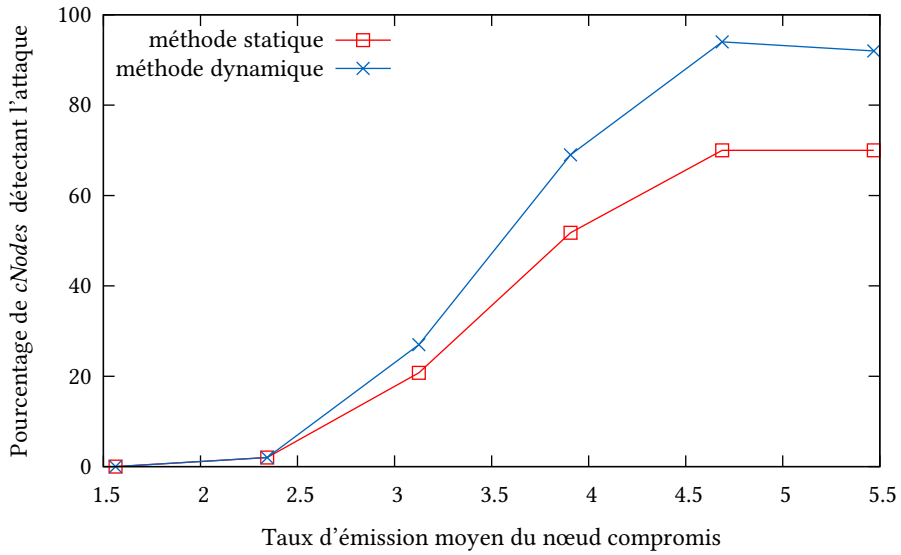

FIG. 4.8 : Taux de détection en fonction de λ_c

2.3 Consommation énergétique

L'étude de la consommation en énergie de la méthode proposée est indispensable pour apprécier l'effet produit sur la batterie des capteurs. Les résultats numériques présentées dans cette sous-section ont été obtenus en exécutant une nouvelle série d'instances, dont les paramètres de simulation sont synthétisés en table 4.3.

L'histogramme donné en figure 4.9 expose la consommation en énergie des nœuds, pour la méthode d'élection statique et pour la méthode dynamique, en fonction du pourcentage de *cNodes*. Par « consommation en énergie », on entend ici la consommation moyenne en fin de simulation ramenée sur la totalité des nœuds, à l'exception du cluster head ainsi que du nœud compromis, qui consomment évidemment plus que les autres, et dont le comportement est invariant entre les méthodes statique et dynamique. Pour rappel, les nœuds normaux (non *cNodes*) ne reçoivent pas les messages de leurs voisins et donc ne consomment pas d'énergie en dehors de leur période d'émission (voir le fonctionnement détaillé de LEACH au chapitre 2, sous-sous-section 2.2.3). La consommation moyenne en fin de simulation est identique pour les deux méthodes : il s'agit d'un résultat attendu, puisque les deux méthodes mettent en œuvre, à tout instant t, autant de nœuds normaux et de *cNodes* l'une que l'autre.

TAB. 4.3 : Paramètres de simulation

PARAMÈTRE	VALEUR
Durée de la simulation	500 secondes
Nombre de capteurs	100
Consommation énergétique en réception	0,394 W
Consommation énergétique en émission	0,660 W

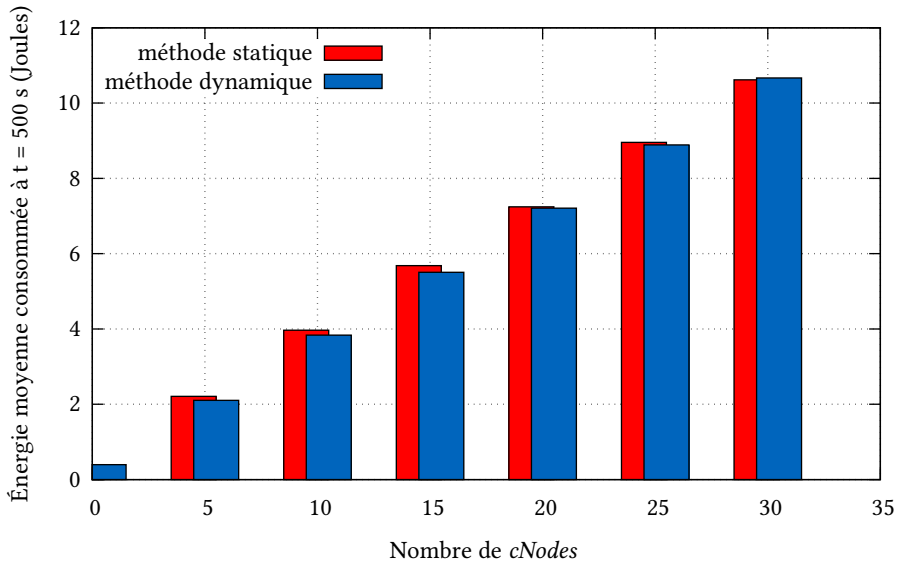

FIG. 4.9 : Consommation moyenne en énergie

En revanche, si l'on compare les écarts-types à la moyenne, on relève des différences notables. Elles sont manifestes sur l'histogramme présenté en figure 4.10, qui présente, pour différents pourcentages de *cNodes*, les écarts-types à la moyenne précédente de la consommation en énergie des nœuds, à la fin de la simulation. Ici encore, cluster head et nœud compromis ont été laissés de côté. Nous observons des écarts-types bien plus forts pour la méthode statique : seuls les *cNodes* initiaux (non soumis

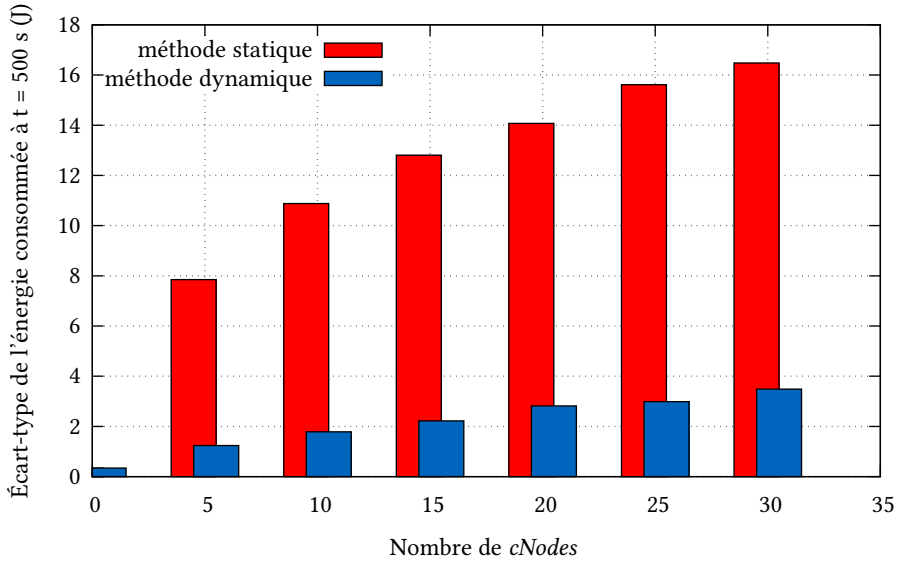

FIG. 4.10 : Écart-type à la moyenne de la consommation en énergie

à réélection par la suite) ont connu une consommation importante — ils sont restés sur écoute, ont analysé le trafic, éventuellement averti le cluster head d'une attaque. Avec la méthode dynamique cependant, les écarts de consommations sont lissés sur le temps du fait des réélections périodiques. Nous observons ainsi une meilleure répartition de la charge.

L'histogramme de la figure 4.11 présente l'instant de décès du premier nœud à épuiser sa batterie au sein du cluster simulé. Pour obtenir ce résultat, nous avons alloué une énergie initiale de 4 J à chacun des nœuds (excepté le nœud compromis et le cluster head, qui ont reçu davantage d'énergie, car nous ne voulions pas qu'ils cessent de fonctionner durant la simulation). À noter : la valeur de 4 J initialement allouée aux nœuds est une valeur très faible. Mais de même, cinq-cents secondes de simulation sont très courtes au regard de la durée de vie réelle d'un capteur. Une pile au lithium (de modèle CR 1225 par exemple) offre environ 540 J ; une pile AA (ou LR06) fournit quant à elle environ 15 400 J, et une pile AAA (ou LR03) environ 6 800 J. À titre d'exemple, plusieurs modèles de capteurs utilisent un couple de piles AAA.

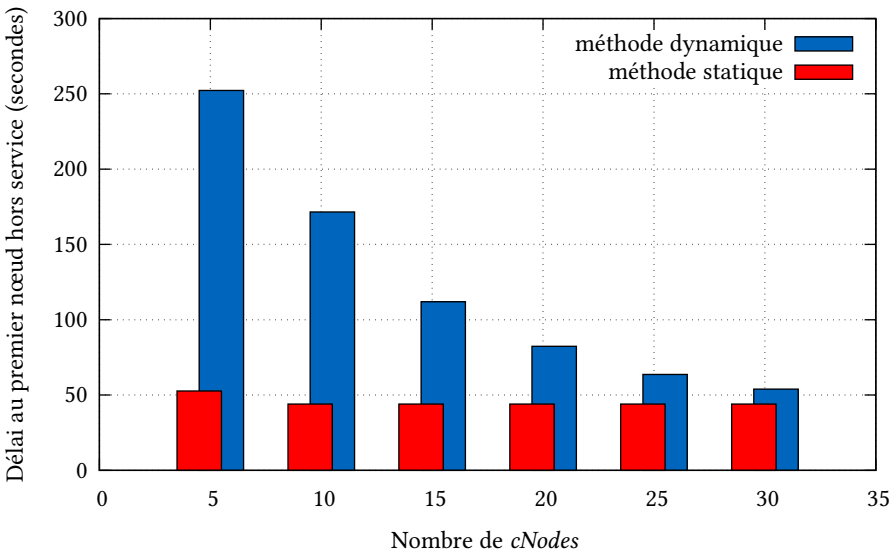

FIG. 4.11 : Instant de premier décès dans le réseau

C'est donc avec 4 J en réserve qu'ont démarré les capteurs, de sorte que l'on puisse effectivement observer les pannes. Nous avons démarré la simulation, et relevé l'instant de « décès » du premier nœud vidé de ses réserves d'énergie. En raison du changement périodique de *cNodes*, la méthode dynamique permet de mieux répartir la consommation en énergie, là où la méthode statique sollicite toujours les mêmes *cNodes*. Le premier nœud à épuiser sa batterie a donc une espérance de vie bien plus longue avec la méthode dynamique (jusqu'à cinq fois plus longue lorsque le pourcentage de *cNodes* est faible), comme le reflète l'histogramme.

2.4 Nœuds restants et détection au fil du temps

Au cours de cette dernière simulation, nous avons observé le nombre de nœuds toujours « en vie », ainsi que le nombre de nœuds détectant le nœud compromis au cours du temps. La durée de la simulation a été étendue à une heure, et les nœuds se sont vus allouer une réserve initiale d'énergie égale à 10 J. Dix pour cent d'entre eux, soit dix à la fois, ont été désignés en tant que *cNodes*. Ces données, et les autres paramètres utilisés, sont présentés en table 4.4.

Tab. 4.4 : Paramètres de simulation

Paramètre	Valeur
Durée de la simulation	3 600 secondes
Nombre de capteurs	100
Pourcentage de *cNodes*	10 %
Consommation énergétique en réception	0,394 W
Consommation énergétique en émission	0,660 W
Quantité initiale d'énergie	10 J

La courbe de la figure 4.12 montre l'évolution dans le temps du nombre de capteurs en vie. Comme précédemment, les capteurs normaux consomment très peu par rapport aux *cNodes*, qui vident leur batterie bien plus rapidement. En conséquence, les dix *cNodes* élus au départ avec la méthode statique consomment leur énergie, et meurent (vers $t = 150$ secondes), tandis que les nœuds normaux vivent beaucoup plus longtemps (environ cinq heures selon nos estimations), mais ne sont plus surveillés. Avec la méthode dynamique, les *cNodes* changent, et le premier nœud à mourir décède bien plus tard que dans le cas précédent. Il s'agit d'un nœud qui a été élu *cNode* sur plusieurs périodes — mais pas (à la différence de la méthode statique) en continu sur *toutes* les périodes. Deux nœuds seulement sont tombés à court d'énergie à $t = 700$ secondes avec la méthode dynamique. En revanche, passé ce stade, les autres nœuds vont rapidement dépérir les uns après les autres ; il ne reste plus qu'un nœud en vie à la fin de l'heure de simulation (ceci n'est pas visible sur la courbe). Ce décès « prématuré » des nœuds pour la méthode dynamique est tout à fait normal et prévisible, puisque la réélection de *cNodes* élève continuellement la consommation de nouveaux nœuds ; tandis que pour la méthode statique, une fois les *cNodes* décédés, il n'y a plus que des nœuds normaux, qui consomment beaucoup moins d'énergie.

Il est alors intéressant d'observer combien de nœuds sont toujours non seulement en vie, mais aussi capables de détecter l'attaque du nœud compromis au cours du temps. Les résultats sont exposés par la courbe donnée en figure 4.13. Le nombre de *cNodes* ayant détecté l'attaque est indiqué sur chaque période de soixante secondes. Après la quatrième minute de simulation, tous les *cNodes* élus par la méthode statique ont stoppé leur activité par manque d'énergie, et les attaques ne peuvent plus être détectées. Avec la méthode dynamique, une moyenne approximative de 6,5 nœuds en vie sur 10 détecte le nœud compromis au cours de chaque période de 60 secondes séparant deux élections. Le capteur malveillant est toujours détecté par plus d'un nœud en moyenne après une demi-heure de simulation.

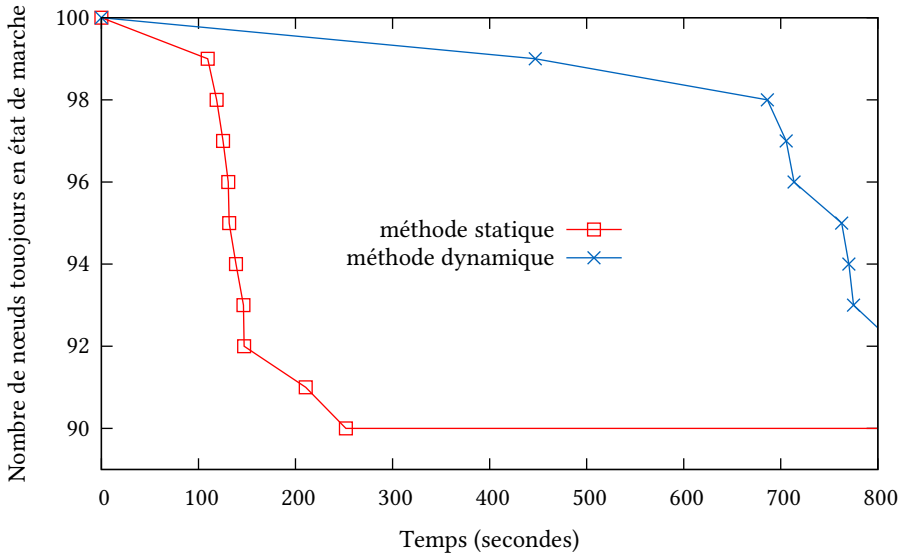

FIG. 4.12 : Nombre de capteurs toujours en vie

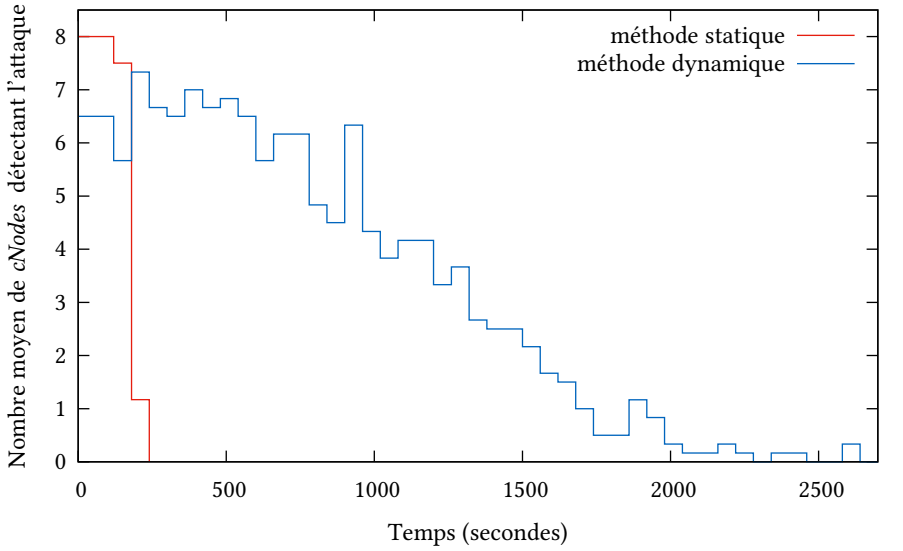

FIG. 4.13 : Détection des attaques de déni de service

3 Modélisation

Nous avons établi un processus de sélection aléatoire, périodiquement renouvelé, des *cNodes* assurant la détection des nœuds compromis dans le réseau. Les simulations réalisées ont fourni des valeurs numériques permettant de valider le modèle proposé ; mais elle n'ont pas la rigueur des méthodes formelles et n'offrent pas, comme ces dernières, la possibilité de vérifier de façon sure des propriétés sur l'état du système.

Dans cette section sont introduits plusieurs modèles pouvant permettre de représenter notre système, et d'en évaluer les performances selon divers critères. Ils peuvent par ailleurs être repris pour modéliser tout système approchant.

3.1 Chaines de Markov

3.1.1 ▶ Présentation et hypothèses du modèle

En premier lieu, nous avons modélisé notre solution sous la forme d'un processus de Markov à temps continu (CMTC, pour chaine de Markov à temps continu). Notre modèle représente un cluster d'un réseau de capteurs sans fil qui vérifie les hypothèses suivantes :

- seul le cluster est modélisé ;
- le cluster contient exactement un CH ; s capteurs chargés d'effectuer des mesures ; et m nœuds jouant le rôle de *cNodes* [2] ;
- il existe exactement un nœud compromis parmi les s capteurs ;
- le capteur i ($1 \leq i \leq s$) génère du trafic selon un processus de Poisson de paramètre λ_i ;
- le nœud compromis c génère du trafic selon un processus de Poisson de paramètre λ_c (tel que $\forall i \in [\![1; s]\!] \backslash \{c\},\ \lambda_c \gg \lambda_i$) ;
- chaque *cNode* effectue une détection du trafic environnant de façon périodique, selon une distribution exponentielle de paramètre μ ; si un trafic anormal est observé, le *cNode* transmet un rapport d'anomalie au CH ;
- la topologie du cluster correspond à un graphe connexe : chaque nœud peut atteindre directement chacun des autres nœuds du cluster.

3.1.2 ▶ Modélisation

Sous les hypothèses précédentes, notre modèle peut être représenté par une CMTC multidimensionnelle de taille m. Il est alors exprimé sous la forme d'un m-uplet $x = (x_1, x_2, \ldots, x_m)$ de macro-états $x_k = (x_{k_1}, x_{k_2}, \ldots, x_{k_s}, x_{k_d})$. Chaque macro-état x_k représente le nombre de paquets détectés par le *cNode* k ($1 \leq k \leq m$). Plus exactement, chaque x_{k_i} ($1 \leq i \leq s$) est un compteur, initialisé à 0, qui est incrémenté à chaque fois que le *cNode* k détecte un paquet en provenance du capteur i. x_{k_d} est une variable de type booléen, initialisée à 0, et passée à 1 lorsque le *cNode* k détecte un

[2] s pour *sensing node*, m pour *monitoring node*.

trafic anormalement élevé. La fonction de seuil $f : s^s \to \{0;1\}$ est utilisée par les *cNodes* pour déterminer la nature normale ou non du trafic (elle indique si un paquet a dépassé ou non le seuil des communications dites « normales »). Elle prend pour argument les valeurs des s compteurs $x_{k_1}, x_{k_2}, \ldots, x_{k_s}$ d'un macro-état x_k associé à un *cNode* k, et la valeur retournée est stockée dans x_{k_d}.

Nous donnons un exemple d'équation de transition pour un macro-état générique $x_k, 1 \leq k \leq m$. Le macro-état x, représentant l'intégralité du système, est directement obtenu en agrégeant les différents x_k. Le compteur x_{k_c} représente dans la suite le nombre de messages émis par le nœud compromis et captés par le *cNode* k. L'équation est donnée en figure 4.14.

$$
\begin{aligned}
x_k \quad &\to \quad \text{Transmission d'un capteur normal} \\
&\to \quad (x_{k_1}, \ldots, x_{k_i} + 1, \ldots, x_{k_c}, \ldots, x_{k_s}, 0) \\
&\qquad\quad \text{avec la fréquence } \lambda_i \neq \lambda_c \\
&\to \quad \text{Transmission du nœud compromis} \\
&\to \quad (x_{k_1}, \ldots, x_{k_i}, \ldots, x_{k_c} + 1, \ldots, x_{k_s}, 0) \\
&\qquad\quad \text{avec la fréquence } \lambda_c \\
&\to \quad \text{Vérification ; détection d'un trafic anormal} \\
&\to \quad (0, \ldots, 0, \ldots, 0, \ldots, 0, 1) \\
&\qquad\quad \text{avec la fréquence } \mu \cdot 1_{f(x_k) \geq \textit{seuil}} \\
&\to \quad \text{Vérification ; aucun trafic anormal détecté} \\
&\to \quad (0, \ldots, 0, \ldots, 0, \ldots, 0, 0) \\
&\qquad\quad \text{avec la fréquence } \mu \cdot 1_{f(x_k) < \textit{seuil}}
\end{aligned}
$$

FIG. 4.14 : Exemple d'équation de transition du processus de MARKOV à temps continu modélisant le cluster

Formulé autrement : le *cNode* k, dans l'état x_k, reçoit périodiquement les messages de chaque capteur i (pour tout i tel que $1 \leq i \leq s, i \neq k$) : ce sont des transmissions dites « normales », qui surviennent avec une fréquence moyenne λ_i. Chaque message reçu laisse le *cNode* k dans un état presque identique, à la seule différence que le compteur x_{k_i} est incrémenté de 1. La même chose se produit avec une fréquence λ_c pour les messages envoyés par le nœud compromis, qui provoque à chaque fois l'incrémentation du compteur x_{k_c}. Enfin, la vérification du trafic enregistré par le *cNode* k survient avec une fréquence μ. La fonction f est appliquée sur l'ensemble des x_{k_i}, i allant de 1 à s. Si l'un des capteurs a envoyé un nombre de paquets supérieur à la valeur de seuil (c'est-à-dire si l'on a $f(x_k) \geq \textit{seuil}$), le drapeau x_{k_d} est levé. Un message d'alerte est alors envoyé du *cNode* k au cluster head. Si, en revanche, aucune anomalie n'est à rapporter, le drapeau x_{k_i} est laissé à 0, et rien n'est envoyé. Dans les deux cas, tous les compteurs x_{k_i} sont réinitialisés à 0, afin de s'assurer que la vérification suivante soit effectuée sur des données « fraiches » (et pour ne pas déclencher par erreur une fausse détection par cumul des nombres de paquets envoyés d'une période sur l'autre).

La probabilité de détection du nœud compromis par le *cNode* k, notée Δ_k, est la suivante :

$$\Delta_k = \sum_{x_{k_1},\ldots,x_{k_s}}^{\infty} \pi(x_{k_1},\ldots,x_{k_s},x_{k_d}=1)$$

où la valeur $\pi(x_{k_1},x_{k_2},\ldots,x_{k_s},x_{k_d})$ représente la distribution stationnaire du macro-état $x_k=(x_{k_1},x_{k_2},\ldots,x_{k_s},x_{k_d})$.

3.1.3 ▶ Limites du modèle

Représenté sous la forme d'un processus de Markov à temps continu, le modèle suppose que la période séparant deux vérifications consécutives du trafic par un *cNode* est une variable aléatoire qui suit une distribution exponentielle. Il s'agit là d'une approximation grossière ; en réalité, la vérification intervient à intervalles réguliers (de longueur fixe), voire même en continu. Il est donc nécessaire de choisir d'autres systèmes que les processus de Markov pour modéliser notre solution avec plus de précision. Nous allons ainsi nous orienter vers des processus non-markoviens, qui ne présentent pas cette contrainte.

3.2 Réseaux de Petri

Une autre représentation possible du système passe ainsi par l'utilisation de processus stochastiques à évènements discrets (PSED, en anglais *Discrete-Event Stochastic Processes*, ou encore *Generalized Semi-Markov Processes* pour « processus semi-markoviens généralisés »). Leur avantage principal sur les processus de Markov est leur capacité à représenter des évènements dont la distribution n'est pas nécessairement exponentielle. Plus particulièrement, nous allons utiliser des réseaux de Petri stochastiques généralisés étendus.

3.2.1 ▶ Réseaux de Petri stochastiques généralisés étendus

Les réseaux de Petri stochastiques généralisés (RPSG) forment une classe de réseaux de Petri destinée à modéliser des processus stochastiques. Leur version étendue (RPSGe) permet de représenter des transitions dont les délais sont distribués de façon quelconque [5]. Il s'agit d'un langage de haut niveau permettant de représenter des processus stochastiques à évènements discrets, qui au contraire des processus markoviens « classiques » ne sont pas limités à la représentation d'évènements à distribution exponentielle. RPSGe reprend, à quelques différences près, les définitions, la syntaxe et la représentation graphique des réseaux de Petri classiques, enrichis par des propriétés supplémentaires portant sur les transitions. Ces dernières sont dites soit *immédiates*, soit *minutées*. Elles sont de plus caractérisées par les éléments suivants :

1. une distribution qui détermine de façon aléatoire le délai écoulé avant que la transition ne soit franchie ;
2. une priorité qui établit de façon déterministe la transition à franchir en premier, en cas d'égalité ;
3. un poids qui est utilisé pour tirer de façon aléatoire la transition à franchir en premier, en cas de conflits sur le délai écoulé et sur les priorités.

Nous utilisons deux types de transitions minutées :

- les distributions minutées exponentiellement distribuées (marquées dans les figures qui suivent par des rectangles vides, voir par exemple la transition T1 sur l'exemple de gauche de la figure 4.15) ;
- les distributions minutées distribuées de façon déterministe (marquées dans les figures qui suivent par des rectangles bleus, voir par exemple la transition T1 sur l'exemple de droite de la figure 4.15).

Les transitions immédiates sont marquées par des rectangles pleins et noirs (voir par exemple la transition T2 dans les exemples de la figure 4.15). Nous avons aussi recours à des arcs inhibiteurs. Il s'agit d'arêtes dont la flèche sur les représentations graphiques est remplacée par un cercle (par exemple, l'arc reliant la place P1 à la transition T2 dans l'exemple de droite de la figure 4.15), et accompagnées d'un chiffre. Les règles de franchissement des transitions sont légèrement différentes avec ces arcs, puisque ces franchissements ne peuvent être effectués que si le nombre de jetons en amont de la transition est strictement inférieur au nombre indiqué sur l'arc. Par exemple, sur la partie droite de la figure 4.15, la transition T2 peut être franchie, car P1 ne contient que deux jetons (on a bien $2 < 3$).

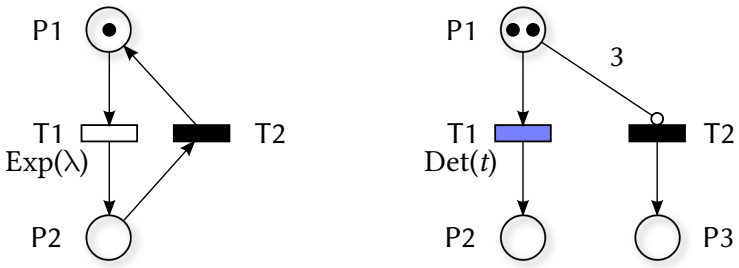

FIG. 4.15 : Exemples simples de RPSGe : transitions immédiates, transitions minutées et arcs inhibiteurs

3.2.2 ▶ Modélisation des nœuds

Nous allons procéder par étapes en modélisant :

1. les nœuds capteurs « simples » ;
2. les *cNodes* désignés de manière statique ;
3. les *cNodes* désignés de façon dynamique (dont la modélisation diffère légèrement).

Ces différentes « briques » pourront être assemblées par la suite pour représenter l'intégralité du système étudié.

Modélisation des capteurs simples Le comportement des capteurs basiques (qui poursuivent simplement leur mission de récolte de données) est simple : chaque capteur i envoie des paquets selon un processus de Poisson de paramètre λ_i. Ce comportement est modélisé par le RPSG présenté en figure 4.16. Le nœud y est symbolisé par les pointillés bleus. Dans ce nœud i se trouve une unique transition TX, sans place en entrée, autrement dit : toujours active. Elle est minutée, avec une distribution exponentielle de paramètre λ_i : chaque évènement correspond à l'envoi de jetons sur tous ses arcs sortants. Ces arcs sont rattachés à la place InBuff_j [3] de chacun des nœuds voisins du nœud i (on trouve donc autant d'arcs sortants que de nœuds voisins). L'intégralité de la fonction de mesure et d'envoi des données utiles du réseau de capteurs peut être modélisée par plusieurs éléments représentés par ce « modèle de nœud ». Un nœud compromis c sera également représenté selon le même modèle, avec un paramètre λ_c plus élevé que les autres λ_i.

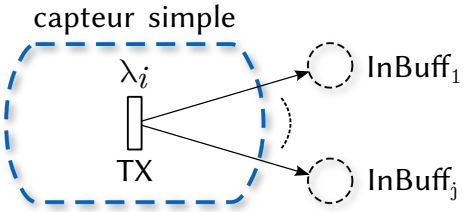

Fig. 4.16 : Modèle RPSG d'un nœud capteur simple

Modélisation du cluster head Nous travaillons de l'intérieur du cluster, et le cluster head est vu comme un simple puits qui reçoit les paquets émis par les autres nœuds. Son activité de retransmission à destination de la station de base n'est donc pas représentée. Illustré en figure 4.17, le CH ne contient donc qu'une place $\text{InBuff}_{\text{CH}}$ qui représente le tampon contenant les messages reçus, et reliée aux transitions TX_i de tous les autres nœuds. En réalité, il devrait y avoir une place InBuff_i distincte pour les jetons provenant de chaque nœud voisin i du cluster head. Pour simplifier la représentation, tous les jetons arrivant au modèle du nœud sont rassemblés dans une place unique.

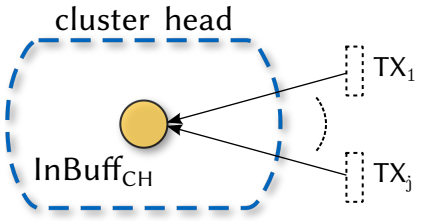

Fig. 4.17 : Modèle RPSG du cluster head

[3] Pour *in buffer*.

Modélisation des *cNodes* désignés de façon statique Un *cNode* élu de façon statique joue uniquement son rôle de *cNode* : il écoute le trafic, mais n'effectue pas de mesures sur son environnement. Le modèle RPSGe associé à ce type de nœud est présenté en figure 4.18.

La place InBuff représente le tampon contenant les messages reçus par le nœud. Un jeton arrivant dans cette place (depuis un nœud « normal ») représente un message capté par le *cNode*. La vérification du respect de la valeur seuil K pour le trafic est réalisée à intervalles de temps constants. Les transitions vérif_OUI et vérif_NON sont « activées » au bout de chaque période, et servent à assurer la vérification. S'il y a au moins K jetons dans la place, la transition vérif_OUI est franchie. Un jeton est alors placé dans la place det, et fait en quelque sorte office de drapeau levé pour indiquer la détection d'un trafic anormal. L'arrivée d'un jeton dans la place det déclenche l'envoi d'un message d'alerte au cluster head. Si, en revanche, il y a moins de K jetons dans la place InBuff, aucun jeton n'est produit dans la place det. Dans un cas comme dans l'autre, la place vider tampon reçoit un jeton (soit par vérif_OUI, soit par vérif_NON). Un jeton dans cette place permet de vider le tampon InBuff : la transition e-déb consomme le jeton de la place vider tampon, et un jeton de InBuff. Un nouveau jeton est créé dans vider tampon.

Le processus est répété jusqu'à ce que la place InBuff soit vide ; dans ce cas, la transition e-fin est franchie et ne produit rien, mais met fin à la consommation des jetons. Les étapes permettant de vider le tampon ne consomment pas de temps, et permettent de repartir avec un compteur neutre pour la période qui mènera à la vérification suivante (de façon à ne pas sommer le nombre de messages perçus d'une période sur l'autre).

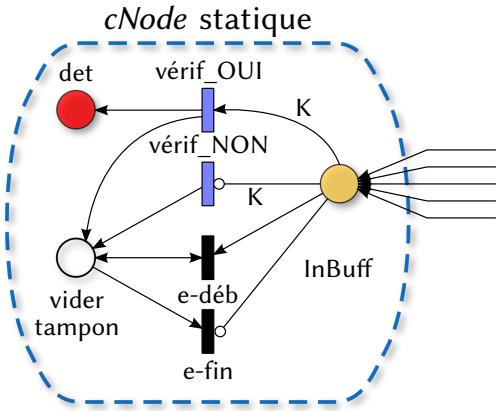

FIG. 4.18 : Modèle RPSGe d'un *cNode* élu de manière *statique*

Modélisation des *cNodes* désignés de façon dynamique Le modèle RPSGe représentant le *cNode* élu de façon dynamique ressemble à son homologue élu de façon statique. Cependant, en raison de l'élection dynamique, tous les nœuds vont assumer périodiquement le rôle de *cNode* et le rôle de capteur simple. Il est donc nécessaire d'ajouter, aux places et transitions des précédents *cNodes*, une place cNode ainsi qu'une transition toujours autorisée, associée à une distribution exponentielle. La première permet d'indiquer à chaque instant si le nœud joue le rôle de *cNode* (si la place contient un jeton) ou bien de capteur simple (cas contraire). La seconde permet de produire des jetons selon un processus de Poisson lorsque le nœud fait office de capteur simple (dans ce cas la place cNode est vide et l'arc inhibiteur autorise le franchissement de la transition TX ; le nœud se comporte exactement comme avec le modèle de la figure 4.16). Le modèle est représenté sur la figure 4.19.

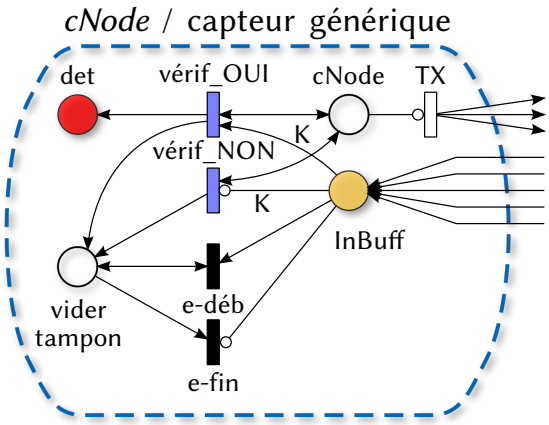

Fig. 4.19 : Modèle RPSGe d'un *cNode* élu de manière *dynamique*

3.2.3 ▶ Modélisation du cluster

***cNodes* désignés de façon statique** Maintenant que la représentation des nœuds individuels a été donnée, nous sommes en mesure de modéliser notre cluster. Pour simplifier la représentation graphique, nous choisissons de représenter un cluster comportant un total de neuf nœuds placés sur une grille de dimensions 3×3. Parmi ces nœuds, deux exactement jouent le rôle de *cNode* (les nœuds 3 et 4 dans notre cas), et un exactement (le nœud 1) est un capteur compromis (distinct des *cNodes*). Le nœud 5 quant à lui tient le rôle du cluster head. La représentation du cluster est donnée en figure 4.20. Également dans l'optique de simplifier la représentation graphique, les transitions e-déb et e-fin, ainsi que la place vider tampon, ont été remplacées par une boite blanche associée à la fonction de vidage de la place InBuff des nœuds. Cette représentation peut être facilement étendue pour modéliser le réseau tout entier. On observera, par rapport au modèle précédent basé sur les chaines de Markov, que le graphe du cluster n'est plus connexe : les nœuds ne peuvent atteindre que leurs voisins directs (sur le côté ou en diagonale sur la représentation). Là encore, il s'agit d'un choix permettant de simplifier le graphique, et la connexité du réseau peut être complétée par le simple ajout des arêtes manquantes.

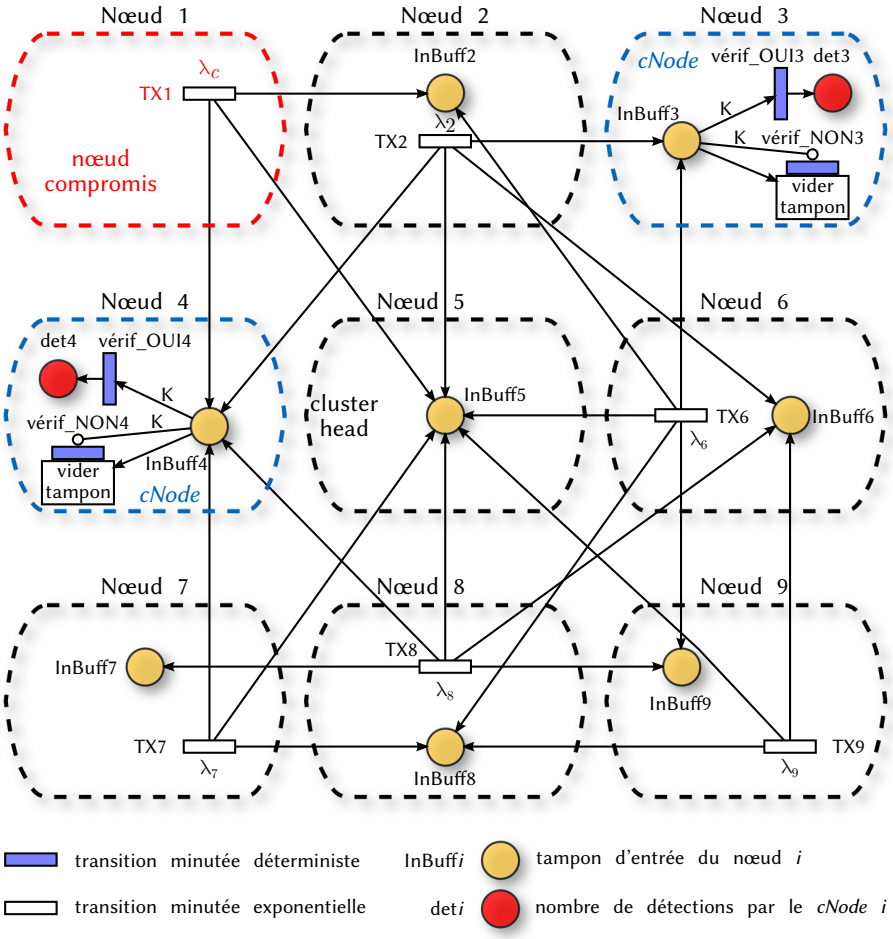

FIG. 4.20 : Modèle RPSGe d'un cluster (topologie : grille 3×3) comprenant un nœud compromis et deux *cNodes* statiques

***cNodes* désignés de façon dynamique** L'élection dynamique des *cNodes* vient compliquer légèrement la modélisation de notre cluster. Bien évidemment, tous les nœuds seront représentés selon le modèle ambivalent illustré par la figure 4.19. Mais il est aussi nécessaire de rajouter un module qui permettra l'élection des *cNodes* pour chaque période. Pour une meilleure clarté, le cluster sera représenté en deux temps.

Tout d'abord, la figure 4.21 représente le cluster comme nous l'avons vu jusqu'ici, sous formes de nœud distincts et de transitions représentant le trafic émis. Les sept nœuds non compromis peuvent endosser alternativement les rôles de capteur simple et de *cNode* (leur représentation a été simplifiée par rapport à la figure 4.19). Le cluster head n'est toujours considéré que comme un simple puits, uniquement équipé d'un tampon mémoire d'entrée pour recevoir les données.

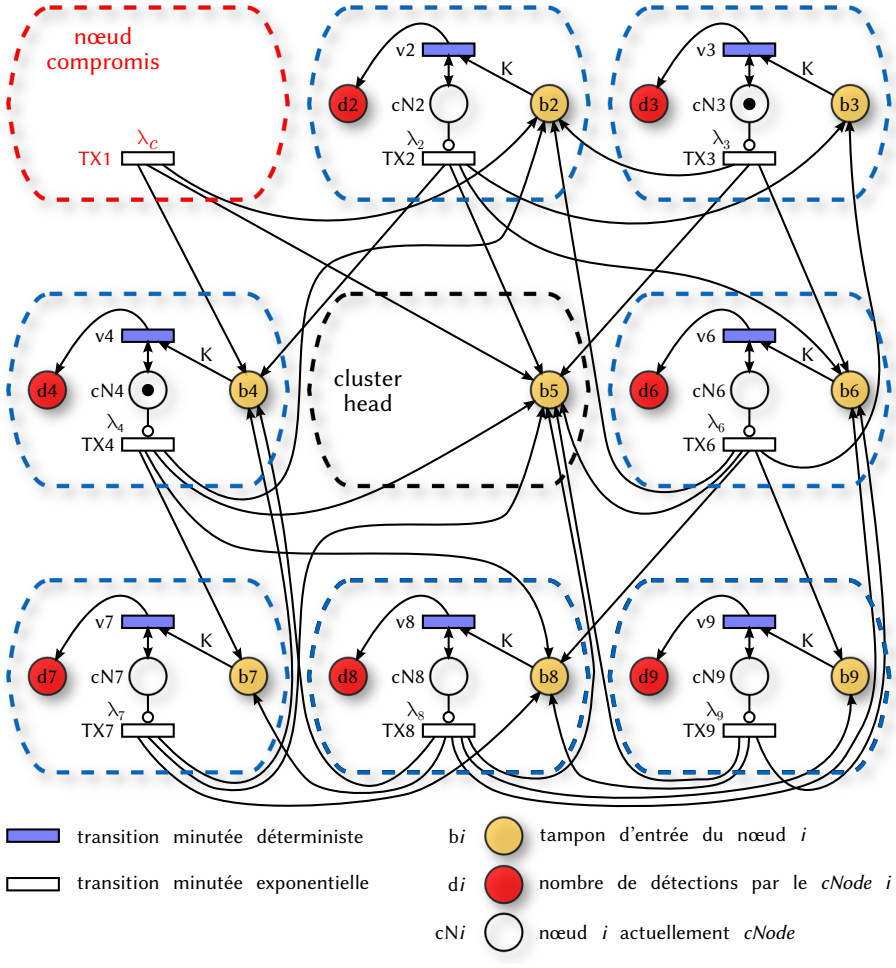

FIG. 4.21 : Modèle RPSGe pour le trafic d'un cluster (topologie : grille 3×3) comprenant un nœud compromis et deux *cNodes* sélectionnés de façon dynamique

Un second réseau de Petri est utilisé pour modéliser le processus de sélection lui-même. En considérant n comme le nombre de résultats possibles à l'élection, le module représentant l'élection dynamique des *cNodes* comprend :

- une unique place centrale ;
- n transitions minutées, suivant une distribution déterministe, et mutuellement exclusives, menant à cette place (en bleu sur la figure 4.22) ;
- n transitions immédiates et mutuellement exclusives, franchissables depuis la place centrale (en noir sur la figure 4.22).

Dans notre cas, pour choisir deux *cNodes* au sein du cluster, et en supposant que le nœud compromis ne peut être désigné, nous avons $n = \binom{7}{2} = 21$ issues possibles pour l'élection. À la fin d'une période d'activité des *cNodes*, une nouvelle élection est déclenchée. La transition minutée correspondant aux *cNodes* sélectionnés pour la période qui s'achève est déclenchée ; les jetons présents dans les places cNode de ces

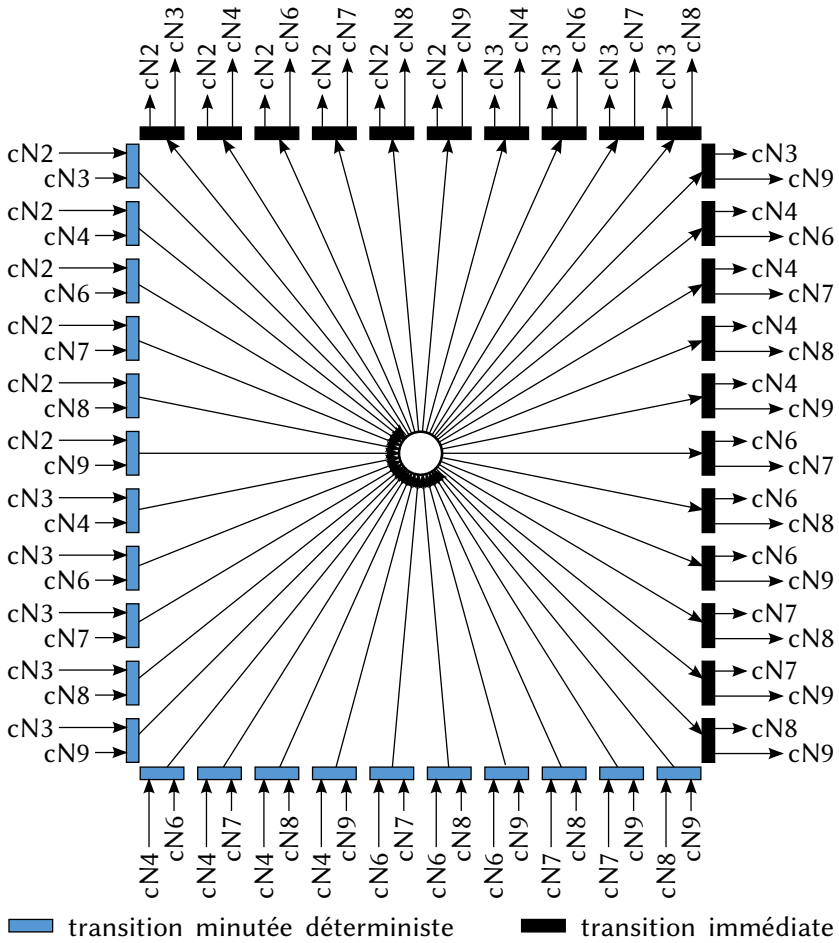

Fig. 4.22 : Modèle RPSGe pour l'élection dynamique de deux *cNodes* parmi les nœuds du cluster

cNodes sont consommés, et produisent ainsi un jeton dans l'unique place centrale du module. Chaque transition immédiate tente alors de produire un jeton. Comme elles sont mutuellement exclusives, mais ont la même priorité et le même poids (voir sous-sous-section 3.2.1), un tirage aléatoire a lieu pour déterminer l'unique transition qui est franchie et qui produit ses jetons. Ceux-ci sont envoyés dans les places cNode des nœuds sélectionnés pour la période qui débute, activant ainsi leurs fonctionnalités de *cNodes*.

3.3 Logique stochastique avec automates hybrides

3.3.1 ▶ Présentation de la logique stochastique avec automates hybrides

La logique stochastique avec automates hybrides (LSAH ; *Hybrid Automata Stochastic Logic* en anglais) est un langage récent, qui introduit une architecture regroupant des études de *model checking* et d'évaluation des performances ainsi que de fiabilité sur des processus stochastiques à évènements discrets (PSED) exprimés sous forme de réseaux de Petri stochastiques généralisés (RPSG) [11]. Plus concrètement, à partir d'un modèle RPSG, les mesures de performances sont exprimées à l'aide de formules LSAH avant que leur soient appliquées des fonctions de *model checking* statique permettant de vérifier automatiquement ces formules.

Pour bien comprendre le fonctionnement de LSAH, voici quelques rappels sur le *model checking*. Il s'agit d'une procédure de vérification formelle utilisant :

- un modèle M à états discrets ;
- une propriété formelle exprimée par une formule de logique temporelle ϕ.

Un algorithme est alors utilisé pour déterminer automatiquement si ϕ vérifie M (ce que l'on note $M \models \phi$). Dans le cas d'un modèle stochastique, des probabilités sont associées aux formules. Vérifier que $M \models \phi$ revient alors à déterminer la probabilité de la formule ϕ dans le contexte du modèle M. LSAH étend ce concept dans le sens où l'évaluation d'une formule peut donner n'importe quel nombre réel, et représenter ainsi une probabilité aussi bien que toute autre mesure de performance. À cette fin, LSAH fait appel à des automates linéaires hybrides (ALH). Un ALH résulte, de manière résumée, de la généralisation des automates temporels dont les variables « horloges » sont remplacées par des variables de données réellement évaluées. Basée sur ce modèle une formule LSAH comprend ainsi deux sous-éléments :

- un ALH permettant la sélection des exécutions temporelles à retenir à partir du modèle RPSG considéré, cette sélection étant le fruit de la synchronisation entre une exécution générée par le modèle RPSG et l'ALH ;
- une expression Z, représentant la mesure à évaluer, construite à l'aide des variables de l'ALH selon la syntaxe suivante :

$$
\begin{aligned}
Z &::= E(Y) \mid Z + Z \mid Z \times Z \\
Y &::= c \mid Y + Y \mid Y \times Y \mid Y/Y \mid \mathrm{dern}(y) \mid \mathrm{min}(y) \\
&\quad \mid \mathrm{max}(y) \mid \mathrm{int}(y) \mid \mathrm{moy}(y) \\
y &::= c \mid x \mid y + y \mid y \times y \mid y/y
\end{aligned}
$$

Cette expression peut être interprétée comme suit :

- x est une variable de données de l'automate :
- y est une expression arithmétique à partir de ces variables de données ;
- Y est une variable de chemin aléatoire, c'est-à-dire une variable évaluée à l'aide d'un chemin de synchronisation, lui-même créé par la synchronisation entre une trajectoire du modèle RPSG et l'ALH ;
- $\text{dern}(y)$ est la dernière des valeurs endossées par l'expression y le long d'un chemin synchronisé ; $\min(y)$ et $\max(y)$ sont les valeurs optimales obtenues le long de ce chemin ; $\text{int}(y)$ est l'intégrale d'y, et $\text{moy}(y)$ la moyenne des valeurs obtenues le long du chemin.

Au final, LSAH fonctionne de la façon suivante :

1. le système est exprimé sous la forme d'un modèle RPSG et d'une formule LSAH ;
2. LSAH génère de façon itérative les trajectoires à partir de l'espace d'états du modèle RPSG et les synchronise avec l'ALH ;
3. les trajectoires acceptées par l'ALH sont évaluées pour obtenir l'estimation de la mesure étudiée. Les trajectoires refusées sont abandonnées.

3.3.2 ▶ Automate linéaire hybride et formule LSAH associés au modèle utilisé

Nous présentons ici quelques exemples de formules LSAH, constitués des expressions LSAH et d'un automate linéaire hybride (ALH) associé au modèle RPSGe présenté plus haut (en figure 4.21). Ces exemples présentent une manière de mesurer des données sur un nœud du réseau de capteurs à partir de la version modélisée sous forme de réseaux de Petri, à fin d'étude, ou bien de comparaison de notre solution avec d'autres systèmes de détection des attaques par déni de service. Ces exemples peuvent en outre être modélisés facilement avec l'outil de *model checking* COSMOS [10].

On considère un nœud quelconque i ($1 \leq i \leq$ nombre de capteurs). L'automate linéaire hybride que nous utilisons fait appel aux variables suivantes :

- x_t : temps global ;
- x_{d_i} : nombre d'attaques détectées par le *cNode* i ;
- x_{TX_i} : nombre de paquets envoyés par le nœud i ;
- x_{bf_i} : nombre de paquets reçus et placés dans le tampon en entrée du nœud i.

L'automate linéaire hybride est présenté en figure 4.23. Il comprend deux états. Ses transitions sont franchies lorsqu'un élément survient (par exemple : égalité de x_t à T, ou franchissement d'une transition sur le modèle RPSGe) ; nous y reviendrons. L'état de gauche, que l'on nommera e_1, contient le taux de variation (autrement dit, la dérivée première) de chacune des quatre variables décrites :

- À chaque unité de temps passée, le temps global (la variable x_t) est incrémenté de $\dot{x}_t = 1$;
- x_{d_i} et x_{TX_i} sont inchangées tant qu'aucune transition du modèle RPSGe n'est franchie. Leurs taux de variation $\dot{x}_{d_i}$ et $\dot{x}_{\text{TX}_i}$ sont nuls ;
- x_{bf_i} est incrémentée avec un taux proportionnel au nombre de jetons présents dans le tampon d'entrée de i (lorsque i joue le rôle de *cNode*). On notera ce taux $\dot{x}_{bf_i}$.

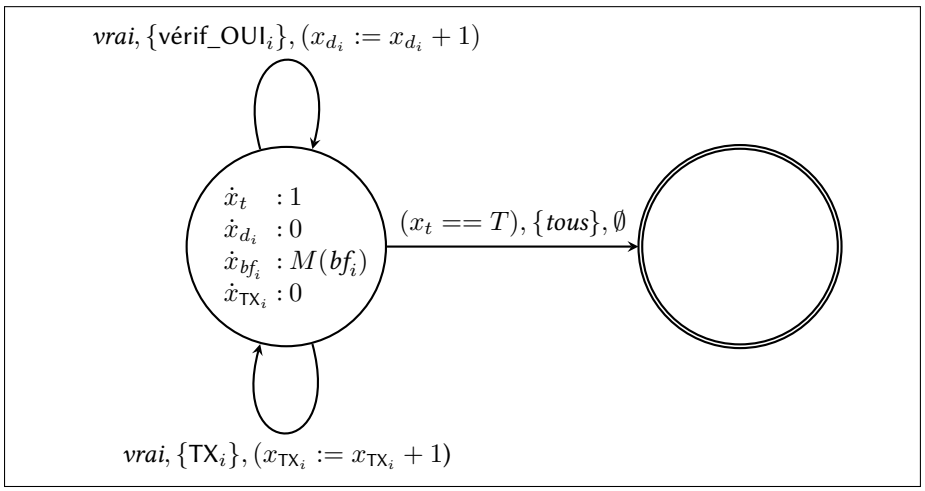

FIG. 4.23 : Exemple d'ALH associé au modèle RPSGe

e_1 comporte également trois transitions, qui sont exprimées sous la forme suivante : <*condition temporelle (sur x_t)*>, <*ensemble des transitions franchies sur le réseau de PETRI pour déclencher l'évènement*>, <*action associée sur les variables de l'automate*>. Il s'agit des transitions suivantes :

- la première transition qui boucle sur e_1 :

$$e_1 \xrightarrow{vrai, \{\text{vérif\_OUI}_i\}, (x_{d_i} := x_{d_i} + 1)} e_1$$

  Elle correspond à l'évènement associé au passage de la transition vérif_OUI du nœud i sur le modèle RPSGe du cluster (présenté en figure 4.21). Lorsque la transition est franchie sur le réseau de PETRI, quelle que soit la valeur de l'« horloge » x_t (condition *vrai*), la transition correspondante sur l'automate linéaire hybride est elle aussi franchie [4]. Comme la transition du réseau de PETRI correspond à une nouvelle détection de trafic anormal par le nœud i, la transition de l'automate a pour effet de mettre à jour la valeur de x_{d_i}, en l'incrémentant de 1 ;

- une deuxième transition qui boucle elle aussi sur e_1 :

$$e_1 \xrightarrow{vrai, \{\text{TX}_i\}, (x_{TX_i} := x_{TX_i} + 1)} e_1$$

  Cette transition fonctionne de façon identique à la précédente, à la différence près que la transition du réseau de PETRI concernée n'est plus vérif_OUI mais TX, et la variable de l'automate est x_{TX_i} : lorsque le nœud i envoie un message (et que TX est franchie sur le modèle RPSGe), la variable x_{TX_i} est incrémentée de 1 ;

- une dernière transition qui mène de e_1 à e_2. Cette transition conduit à l'état final de l'automate, qui accepte et valide le chemin des transitions parcourues

[4] Attention : les transitions du réseau de PETRI, représentées par des rectangles et reliées aux places par des arcs, ne doivent pas être confondues avec les transitions des automates, représentées par des arcs, et reliant les états.

pour l'étudier à l'aide des formules LSAH :

$$e_1 \xrightarrow{(x_t == T), \{tous\}, \emptyset} e_2$$

La transition est autorisée lorsque x_t vaut T, autrement dit : elle est déclenchée toutes les T unités de temps. Elle accepte donc n'importe quel chemin dans l'automate, dont la durée est de T unités de temps. Elle ne dépend pas d'une transition particulière du modèle RPSGe (puisqu'elle les accepte toutes, $\{tous\}$). Aucune action sur les variables n'est effectuée lorsque la transition est franchie.

Une fois un chemin validé par l'automate, il peut être analysé à l'aide de formules LSAH. Voici quelques exemples de formules LSAH permettant d'obtenir plusieurs valeurs sur le cluster considéré à partir des variables définies plus haut pour l'automate :

- $Z_1 \equiv E(\text{dern}(x_{d_i}))$: le nombre attendu d'attaques détectées par le *cNode* i après T unités de temps ;
- $Z_2 \equiv E(\text{dern}(x_{d_i} + x_{d_{i'}}))$: la somme attendue des attaques détectées par les nœuds i et i' après T unités de temps ;
- $Z_3 \equiv E(\text{dern}(x_{\text{TX}_i}))$: le nombre attendu de paquets envoyés par le nœud i après T unités de temps ;
- $Z_4 \equiv E(\text{int}(x_{bf_i}))$: le cumul attendu du nombre de paquets reçus par le nœud i après T unités de temps.

3.4 Amélioration de la solution

Le renouvellement périodique des nœuds de surveillance permet d'améliorer tout à la fois la détection des attaques et la répartition de la charge en énergie dans le cluster, comme le démontrent les valeurs obtenues par simulation. Suite aux résultats numériques, le système a été modélisé de plusieurs façons différentes : après avoir étudié les limites des processus markoviens, orientés vers la modélisation d'évènements à distribution exponentielle, nous avons présenté une autre évaluation formelle possible. Elle fait intervenir une version étendue des réseaux de Petri, dédiée à la représentation de toutes sortes de distributions évènementielles, et permet la visualisation du modèle sous l'aspect classique des réseaux de Petri. Elle est accompagnée d'une troisième méthode basée sur un langage récent, la logique stochastique avec automates hybrides, qui réutilise le modèle précédent pour permettre l'évaluation formelle des performances du système par le biais d'outils de *model checking*. Toutes ont pour objectifs, d'une part, de modéliser notre solution afin d'en souligner les avantages et inconvénients ; mais aussi, d'autre part, de présenter différents moyens existants d'employer les méthodes formelles en vue de modéliser des réseaux de capteurs, et ce de façon plus générale que dans le seul contexte de la solution proposée.

Cette solution, d'ailleurs, a reposé jusqu'à présent sur un processus de sélection aléatoire des *cNodes*, qu'il soit mené au niveau du cluster head, de la station de base, ou des capteurs eux-mêmes. Mais n'y aurait-il pas un moyen plus efficace d'assigner les rôles au nœuds du cluster ? Il devrait être possible d'utiliser d'autres paramètres, comme l'énergie des capteurs, ou la réputation qu'ils ont su entretenir auprès de leurs voisins respectifs, pour établir un processus plus efficace. Pour tenter de répondre correctement à la question, ce changement du mode de sélection fait l'objet des deux chapitres qui suivent.

5

Sélection des *cNodes* selon l'énergie résiduelle

Le chapitre 4 expose un algorithme de sélection pseudo-aléatoire des nœuds de surveillance. L'un des inconvénients de cet algorithme est que l'énergie résiduelle des capteurs, c'est-à-dire le niveau d'énergie restant dans leur batterie, n'est pas mesurée, et surtout n'est pas prise en compte lors du processus de renouvellement des *cNodes*. Pourtant, la surveillance que ces derniers mènent dans leur cluster entraine une consommation énergétique accrue, puisqu'ils doivent demeurer en écoute sur le médium de façon continue.

Dans ce chapitre, une deuxième méthode de sélection des *cNodes* est proposée. Le concept initial de l'algorithme est simple : à chaque renouvellement du processus, l'énergie résiduelle des capteurs est évaluée, et ceux possédant le plus de réserves deviennent *cNodes*. Le but de cette méthode est évidemment d'atteindre une meilleure répartition de la consommation d'énergie dans le cluster : les capteurs possédant le plus haut niveau de charge batterie se verront attribuer le rôle qui consomme le plus d'énergie.

Néanmoins, la perte de l'aspect aléatoire au profit d'une méthode purement déterministe va entrainer un certain nombre de contraintes concernant sécurité et couverture spatiale. Il faut ainsi mettre en place des parades face aux nœuds compromis qui tenteraient de conserver *ad vitam æternam* le rôle de *cNode* en falsifiant leur niveau d'énergie. Un nouveau type de nœuds, les *vNodes*, viennent ici prévenir les tentatives de fraudes, tandis que le cluster head veille à la couverture correcte du réseau par les *cNodes*. Les performances obtenues à l'aide de ce second mécanisme de sélection sont elles aussi étudiées à travers un jeu de simulations.

1 Mécanisme de sélection des *cNodes*

1.1 Utilisation des *vNodes* pour sécuriser le processus de sélection

Les hypothèses présentées en début de chapitre 4 (sous-section 1.1) demeurent valables ici. Le processus de sélection dans ce chapitre repose sur l'énergie restante pour chaque nœud au moment du renouvellement. Cette énergie disponible dans la batterie est dite « énergie résiduelle ». Afin de répartir au mieux la charge, les nœuds possédant le plus d'énergie en réserve sont sélectionnés pour assurer la tâche qui consomme le plus. Mais l'usage de la valeur d'énergie résiduelle comme critère de sélection présente un inconvénient notable : il n'y a aucun agent, dans le réseau, capable de mesurer de façon fiable le niveau d'énergie présent dans la batterie d'un nœud donné, si ce n'est ce nœud lui-même. Les nœuds voisins d'un nœud N peuvent tout au plus enregistrer les messages envoyés par N, et en inférer une valeur approximative de l'énergie consommée. Mais comme ils ne connaissent ni le niveau d'énergie initial de N (au moment du déploiement du réseau) ni l'énergie dépensée par N lors de l'écoute du médium de transmission, la consommation approximative calculée ne permet pas de déduire le niveau d'énergie résiduelle avec une précision suffisante pour classer les nœuds de façon fiable sur ce critère.

Aussi la seule façon d'obtenir le niveau de charge de la batterie d'un nœud est-elle de le lui demander. L'algorithme de sélection proposé s'écrit donc de la manière suivante :

1. Au cours d'une première phase, chaque nœud évalue la valeur de son énergie résiduelle et la transmet au cluster head ;
2. Une fois cette phase terminée (lorsque le cluster head a reçu la valeur de chaque nœud, ou lorsque le délai d'attente a expiré), le CH sélectionne les n capteurs possédant le niveau d'énergie le plus élevé (où n représente le nombre de *cNodes* que l'on souhaite obtenir durant chaque cycle) et envoie à ceux-ci un message de notification pour leur assigner leur rôle de *cNode*.

Cet algorithme est déterministe, et élimine tout aspect aléatoire du processus de sélection. La règle est simple : les n nœuds possédant le plus d'énergie en réserve au moment de la sélection sont désignés. Comme le rôle de *cNode* est censé impliquer une consommation plus élevée en ressources énergétiques, la rotation des rôles est théoriquement assurée.

Mais l'aspect entièrement déterministe du processus peut aussi représenter une faille qui pourrait être exploitée par des nœuds compromis souhaitant s'assurer d'être sélectionnés. Un attaquant a effectivement intérêt à faire attribuer le rôle de *cNode* aux nœuds qu'il a compromis, car cela lui permet :

- de réduire le nombre de *cNodes* légitimes dans le réseau capables de détecter les capteurs compromis ;
- de signaler au cluster head des capteurs « innocents » comme étant compromis, dans le but de les faire exclure du réseau.

Quand un algorithme de sélection (pseudo-)aléatoire est appliqué, un nœud compromis peut très bien être désigné pour un cycle, mais il perdra rapidement son rôle de

surveillance, pour les cycles suivants. Même avec un processus d'auto-désignation (basé sur le modèle de LEACH par exemple), les nœuds compromis pourraient tenter de conserver leur rôle de *cNode* sur plusieurs cycles, mais cela n'empêcherait pas des capteurs légitimes de s'auto-désigner également. Avec un processus déterministe, les nœuds compromis peuvent en revanche monopoliser la plupart des rôles de *cNodes*. Tout ce qu'ils ont à faire, pour en arriver là, consiste à annoncer un niveau de charge batterie plus élevé que celui des capteurs légitimes lors de la première phase de l'algorithme de sélection. Ils sont alors assurés d'être désignés *cNodes* ; et s'il se trouve y avoir au moins n nœuds compromis au sein du cluster, ils peuvent de cette manière accaparer les n rôles de *cNode* et désactiver du même coup le mécanisme de détection dans son intégralité. L'attaque devient alors impossible à détecter et à contrer.

Pour empêcher les capteurs de « mentir » lorsqu'ils annoncent leur valeur d'énergie résiduelle, l'assignation d'un nouveau rôle à certains des voisins de chaque *cNode* est proposée [20, 18]. Ces nœuds, que nous appelleront désormais *vNodes* (comme pour *nœuds de vérification*), seront responsables de vérifier la consommation en énergie des nœuds de surveillance. Une fois que la sélection des *cNodes* est réalisée, chaque voisin d'un *cNode* nouvellement désigné choisit à partir d'un seuil de probabilité prédéfini s'il sera ou non *vNode* pour ce *cNode*. Un capteur peut assumer le rôle de *vNode* pour plusieurs *cNode* différents, du moment qu'il s'agit de nœuds voisins. Les *vNodes* d'un cluster sont représentés sur la figure 5.1.

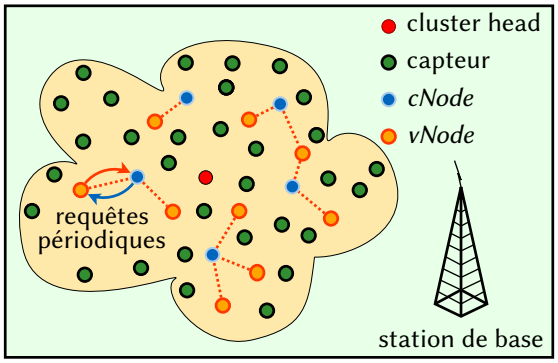

FIG. 5.1 : Schéma : cluster avec *cNodes* et *vNodes*

Si le rôle de *vNode* s'avère trop gourmand en énergie, il devient inutile : autant déployer dès le départ, dans ce cas, un mécanisme de sélection pseudo-aléatoire des *cNodes*. Aussi les *vNodes* ne doivent-ils pas demeurer en permanence à l'écoute du médium, comme le font les *cNodes*. Ils se contentent d'envoyer, de temps en temps, une requête au *cNode* qu'ils surveillent, pour lui demander son niveau d'énergie. Ils conservent la réponse du *cNode* en mémoire.

Une fois qu'ils ont collecté suffisamment de données, les *vNodes* essayent d'établir une corrélation entre le modèle théorique de la consommation du *cNode* surveillé, et les valeurs que celui-ci annonce (valeurs obtenues à la fois lors des phases de sélection et en réponse aux requêtes). Quatre cas sont alors susceptibles de se produire :

1. La consommation en énergie annoncée par le *cNode* ne correspond pas du tout au modèle théorique : il y a alors une forte probabilité pour que le nœud soit corrompu, et cherche à accaparer le rôle de *cNode*. Il est dénoncé au cluster head ;

2. La consommation annoncée correspond très exactement au modèle théorique : le capteur est probablement un nœud compromis qui tente d'être sélectionné tout en échappant à la détection des *vNodes*. Autrement dit : le nœud est compromis, mais il adapte son comportement par rapport au point précédent. Mais il est quasiment impossible en pratique que les valeurs théoriques et empiriques soient exactement identiques, sans la moindre marge d'erreur. Le capteur est donc dénoncé au cluster head ;

3. La consommation correspond à peu près aux valeurs du modèle théorique, mais elle n'évolue pas dans le sens de la consommation réelle observée localement par les *vNodes* (l'évolution locale dans le temps du *cNode* devrait être grossièrement similaire à celle des *vNodes*, puisqu'ils sont voisins, mais ce n'est pas le cas). Il s'agit probablement d'un nœud compromis qui adapte son comportement par rapport aux deux premiers points, par exemple en décrémentant son énergie des montants minimums imposés par la marge d'erreur définie vis-à-vis du modèle théorique. Le nœud est dénoncé au cluster head ;

4. La consommation annoncée correspond grossièrement au modèle théorique, et évolue dans le sens de la consommation locale observée par les *vNodes*. Que le nœud soit compromis ou non, il adopte ici un comportement normal, et est autorisé à endosser le rôle de *cNode*.

Si un *vNode* est compromis, il peut tenter de faire exclure le *cNode* légitime qu'il observe. Le cluster head doit donc recevoir les rapports de plusieurs *vNodes* distincts (selon un seuil prédéterminé) avant de considérer un *cNode* comme étant réellement compromis. Dans une certaine mesure, cette précaution permet aussi de protéger un *cNode* légitime des erreurs (faux positifs) de ses *vNodes*.

Le recours au rôle de *vNode* assure ainsi que seuls les nœuds annonçant un niveau d'énergie résiduelle plausible seront autorisés à agir en temps que *cNodes*. Comme ce dernier rôle consomme plus d'énergie que le relevé et la transmission des mesures sur l'environnement des capteurs, les nœuds sélectionnés pour être *cNodes* verront tôt ou tard leur énergie résiduelle descendre sous le niveau des nœuds assurant leurs fonctions de base ; une fois ce stade atteint, ils sont assurés de perdre leur rôle pour le cycle suivant. On remarquera que les points 2 et 3 énoncés ci-dessus correspondent à un nœud compromis qui décrémente volontairement son énergie annoncée au cours du temps ; même si des irrégularités peuvent être détectées et le nœud compromis être ainsi démasqué, le seul fait qu'il applique ce comportement assure tout de même que le capteur compromis cessera d'être désigné *cNode* pour un cycle ou pour un autre dans le futur.

Le modèle énergétique implémenté par les *vNodes* pour vérifier les affirmations des *cNodes* ne doit pas être trop lourd, mais il doit demeurer suffisamment précis pour pouvoir déterminer si un *cNode* s'en éloigne véritablement, tout en limitant les faux positifs. Nous proposons ici d'utiliser le modèle de diffusion introduit par Rakhmatov et Vrudhula [101]. Ce choix repose sur plusieurs observations :

- Ce modèle fournit une approximation assez précise de la consommation réelle d'une batterie, et tient compte des processus chimiques internes à la batterie tels que les effets de récupération (*recovery effect* en anglais) ou de perte de capacité selon la tension de décharge (*rate capacity effect*) ;

- Il s'agit de l'un des modèles implémentés dans ns, ce qui permet, pour les simulations basées sur ce logiciel, de bénéficier d'une implémentation déjà existante,

et surtout d'un modèle théorique parfait. Il ne faut pas craindre cependant que ce modèle ne soit « trop parfait » pour être utilisé par les *vNodes*, étant donné que ces derniers estimeront l'énergie consommée par les *cNodes* à partir de messages émis par ces derniers, sans connaissance des mécanismes internes au nœud (calcul, écoute) comme le fait le cœur du simulateur. Les valeurs calculées par les *vNodes* demeurent donc approximatives par rapport à la consommation des *cNodes* réellement calculée par ns.

Le modèle de diffusion de Rakhmatov et Vrudhula se réfère aux réactions chimiques qui se produisent à l'intérieur de l'électrolyte de la batterie. Cette diffusion est résumée par l'équation 5.1

$$\sigma(t) = \underbrace{\int_0^t i(\tau)\,\mathrm{d}\tau}_{l(t)} + \overbrace{\int_0^t i(\tau)\left(2\sum_{m=1}^{\infty}\exp^{-\beta^2 m^2 (t-r)}\right)\mathrm{d}\tau}^{u(t)} \tag{5.1}$$

où :

- $\sigma(t)$ est la charge apparente perdue par la batterie à l'instant t ;
- $l(t)$ est la charge « utile » ;
- $u(t)$ est la charge inutilisable (« perdue dans la batterie ») ;
- $i(t)$ est l'intensité du courant électrique à l'instant t ;
- $\beta = \dfrac{\pi\sqrt{D}}{w}$, où D est une constante de diffusion et w la largeur de l'électrolyte.

En pratique, le calcul des dix premiers termes de la somme fournit une bonne approximation — il s'agit par ailleurs du comportement par défaut de ns pour l'utilisation de ce modèle énergétique.

L'intérêt des *vNodes* peut être résumé ainsi : un nœud compromis ne peut pas accaparer indéfiniment le rôle de *cNode* sans tricher lors des annonces, auquel cas ils sont détectés par les *vNodes*. La détection des *cNodes* corrompus, ou bien le fait de les forcer à abandonner leur rôle pour un cycle futur, sont donc les deux objectifs des *vNodes*. Ce rôle n'empêche pas, par ailleurs, le nœud qui l'endosse de poursuivre ses activités de mesure : les requêtes auprès des *cNodes* ne doivent pas survenir trop souvent, sous peine de décharger trop rapidement la batterie des *vNodes*. La machine à états des capteurs est présentée en figure 5.2.

1.2 Couverture du cluster en cas d'activité hétérogène

Le passage d'un algorithme pseudo-aléatoire à un processus de sélection déterministe n'a pas pour seul inconvénient d'introduire une faille que des capteurs compromis pourraient tenter d'exploiter. Il soulève un second problème, illustré en figure 5.3, indépendant du comportement des nœuds, et qui pourrait gêner la détection de capteurs compromis. S'il advient qu'une certaine zone géographique du cluster produit un trafic plus important que le reste des nœuds, alors l'énergie des capteurs de cette zone diminuera plus rapidement. Il s'ensuit mécaniquement que les *cNodes* ne seront pas ou peu sélectionnés parmi les capteurs de cette zone. Il y a même de forte chances

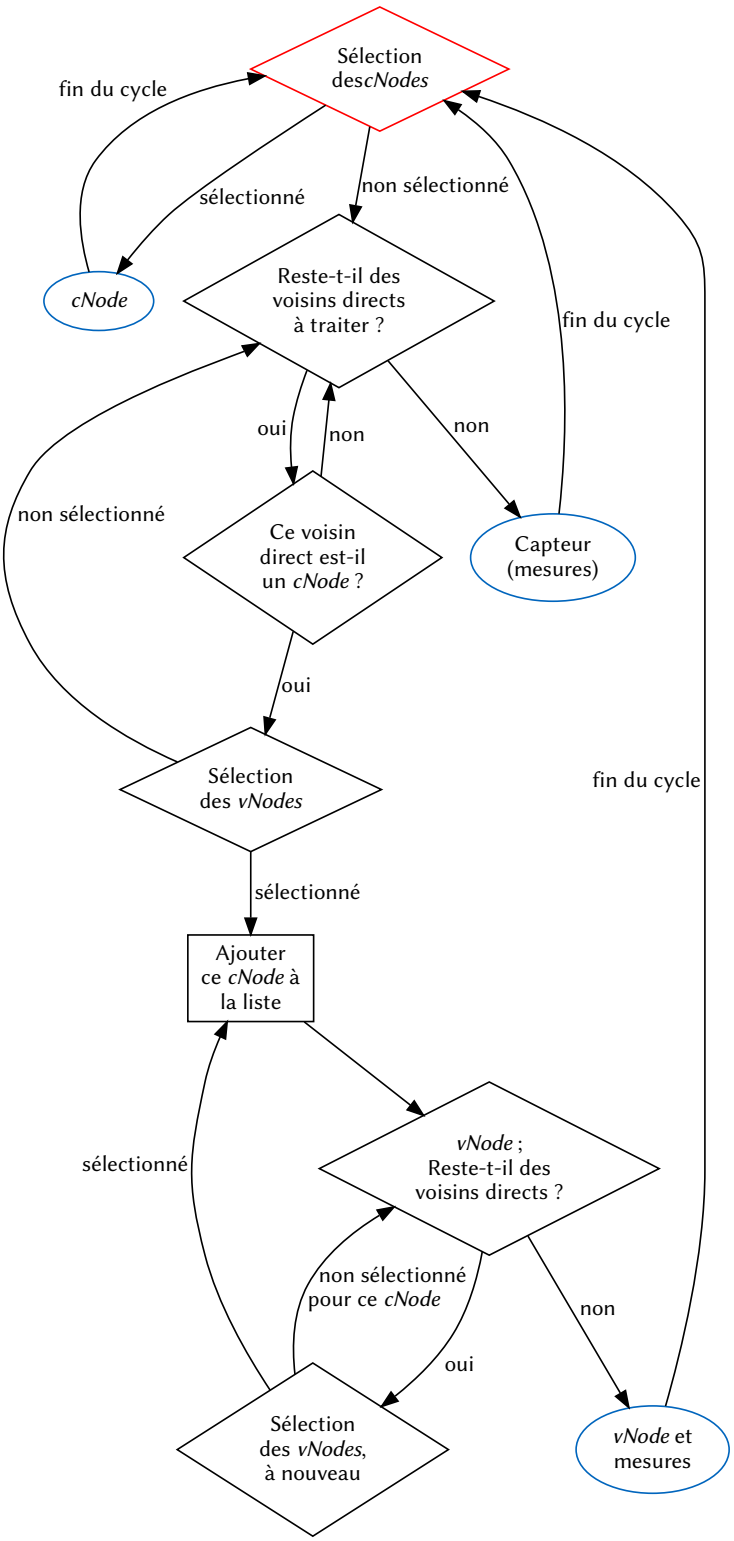

FIG. 5.2 : Machine à états des capteurs (non-*cNode*)

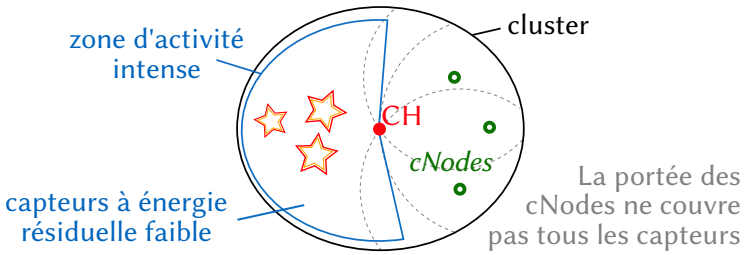

FIG. 5.3 : Schéma d'explication : les *cNodes* sont sélectionnés au sein de la zone où l'activité est la plus faible (donc ou les nœuds ont conservé le plus d'énergie), et leur surveillance ne couvre pas les capteurs situés dans la partie opposée du cluster.

que les n *cNodes* désirés soient sélectionnés en dehors de la zone en question, et que certains capteurs ne soient pas surveillés du tout tant que le trafic de la zone d'activité ne diminue pas, donc potentiellement pour tous les cycles.

Pour assurer une couverture complète du réseau, il est nécessaire de s'assurer que chaque capteur est surveillé par au moins un *cNode*. Aussi l'algorithme de sélection présenté plus haut doit-il être modifié, pour devenir :

1. Au cours d'une première phase, chaque nœud évalue la valeur de son énergie résiduelle et la transmet au cluster head ;
2. Le cluster head écoute toutes les valeurs. Les autres nœuds enregistrent également les valeurs de leurs voisins ;
3. Tous les nœuds envoient au CH la liste de leurs voisins directs (*1-hop*) [1] ;
4. Le CH sélectionne les n *cNodes* parmi les nœuds ayant le plus d'énergie résiduelle, de telle façon que ces n nœuds couvrent tous les autres capteurs par leur portée d'émission ;
5. Au besoin (si n est trop bas), le CH sélectionne des *cNodes* supplémentaires de façon à ce qu'ils couvrent l'intégralité du cluster ;
6. Le CH envoie un message aux nœuds sélectionnés pour leur notifier leur rôle de *cNode*.

Il est intéressant de noter que certains algorithmes de clusterisation (HEED [137] par exemple) utilisent d'autres mécanismes de sélection basés sur l'énergie résiduelle des nœuds — pour choisir des cluster heads, mais qui pourraient être transposés à la sélection des *cNodes*. Nous ne souhaitons pas utiliser dans ce chapitre de telles solutions qui n'utilisent l'énergie que comme l'un des paramètres pris en compte, et reposent donc sur la confiance vis-à-vis des nœuds qui calculent leur score propre. Au contraire, nous préférons obtenir l'énergie annoncée par les nœuds pour la comparer au modèle théorique, et permettre ainsi aux *vNodes* de veiller à la sécurité du processus.

[1] Nous supposons ici que les capteurs ne trichent pas durant cette étape. Ils pourraient en effet annoncer des voisins « virtuels » supplémentaires, pour faire croire qu'ils seront à portée d'un *cNode*, et ainsi éviter d'être détecté si ce n'est pas le cas. Le chapitre 3 présente des mécanismes de protection permettant de détecter ce type d'attaques.

1.3 Observations

Mécanisme de détection La détection des attaques par déni de service est inchangée par rapport au chapitre précédent. Les *cNodes* appliquent un ensemble de règles sur le trafic collecté dans le cluster : lorsqu'un capteur enfreint une règle un certain nombre de fois, par exemple lorsque son débit excède un seuil prédéterminé, il est considéré comme malveillant (voir le paragraphe à ce sujet au chapitre 3, sous-section 2.4.2).

Alourdissement du modèle La solution introduite de par le recours aux *vNodes* consiste littéralement à choisir des nœuds chargés de surveiller les surveillants : le concept de surveillance devient récursif. Le processus de sélection lui-même est alourdi par les messages des nœuds à destination de leur cluster head, annonçant leur énergie résiduelle et la liste de leurs voisins. Il est évident que la complexité de l'algorithme de sélection, et par suite logique celle de l'implémentation du système, s'en retrouvent rehaussées par rapport au mécanisme de sélection pseudo-aléatoire des *cNodes*.

Allègement possible Les *vNodes* sont utilisés pour empêcher les nœuds compromis de conserver un rôle de *cNode* sur plusieurs cycles sans être détectés. Lors du premier cycle où un nœud donné occupe le rôle de *cNode*, il n'y a donc guère d'intérêt à le surveiller : il est statistiquement plus probable qu'il s'agisse d'un nœud légitime plutôt que compromis, et il a de bonnes chances d'abandonner le rôle au cycle suivant. Le rôle de *vNode* pour un *cNode* donné peut donc attendre, pour être assigné, le deuxième cycle consécutif pendant lequel le *cNode* occupe son rôle.

Sécurité du réseau Autre point : l'usage des *cNodes* combinés aux *vNodes* n'apporte pas de garantie totale sur la sécurité du réseau. Le système repose sur plusieurs hypothèses, comme le fait que des nœuds compromis ne collaborent pas ensemble : un capteur compromis sélectionné comme *cNode* pourrait annoncer au cluster head, par exemple, qu'il peut entendre et surveiller une portion plus large du cluster que ce qu'il en est réellement, dans le but de faire croire au CH— à tort — que tout le cluster est surveillé. Si le CH ne désigne aucun autre *cNode* pour surveiller ces zones, d'autres nœuds compromis pourraient y porter des attaques sans risque d'être détectés. Une seconde hypothèse à prendre en compte établit que le cluster head n'est pas et ne peut pas être compromis. Troisièmement, il est nécessaire d'être en mesure de contenir les tentatives d'usurpation d'identité (voir le chapitre 3, sous-section 1.3), sans quoi un nœud compromis n'a qu'à simuler une identité différente à chaque tour, possédant une batterie (soit disant) pleinement chargée, pour obtenir systématiquement le rôle convoité. Mais une nouvelle fois, la sécurité d'un système est avant tout une affaire de compromis : plus on cherche à bloquer d'attaques, plus la complexité augmente.

Outils de modélisation Si aucune représentation formelle du système n'est donnée dans ce chapitre, le système n'en a pas moins fait l'objet d'une modélisation, dans une autre étude [54], sous forme d'automates temporels, dont des propriétés ont été exprimées à l'aide d'un sous-ensemble de CTL (*Computation Tree Logic*, « logique d'arbre de calcul ») pour être vérifiées à l'aide de l'outil de *model checking* UPPAAL [2] [15, 128].

[2] Il ne s'agit pas cette fois d'un sigle, mais d'un nom composé à partir des deux principaux établissements qui ont contribué à l'élaboration de l'outil : le département d'informatique de l'université d'Uppsala (UPP) en Suède, et le département d'informatique de l'université d'Aalborg (AAL) au Danemark.

2 Simulation du processus

2.1 Résultats numériques

Le processus de sélection des *cNodes* selon leur énergie résiduelle a, lui aussi, conduit à la réalisation de plusieurs simulations, afin de le comparer au modèle de sélection pseudo-aléatoire. Le logiciel ns a été utilisé pour l'occasion.

Plusieurs instances de simulation ont été réalisées, avec l'accent porté sur les valeurs de l'énergie consommée ainsi que sur la répartition de la charge dans le cluster. Les mécanismes de détection des deux méthodes sont quasiment identiques, et fournissent donc des résultats similaires à ceux obtenus au chapitre précédent en sous-section 2.2 (sur la méthode avec renouvellement périodique) pour ce qui est du taux de détection des nœuds compromis. Les paramètres utilisés lors des simulations sont présentés en table 5.1.

Tab. 5.1 : Paramètres de simulation

Paramètre	Valeur
Nombre de nœuds	30 (plus 1 CH)
Nombre de *cNodes*	4
Probabilité de sélection d'un *vNodes*	33 %
Période de renouvellement des *cNodes*	1 minute
Durée totale de simulation	30 minutes
Forme du cluster	carré
Taille du cluster	2×50 mètres de diagonale
Portée de transmission	50 mètres
Position des nœuds	CH : au centre ; autres : aléatoire
Mobilité des nœuds	nulle
Débit d'émission des nœuds normaux	1024 octets toutes les 3 secondes
Requêtes des *vNodes* (par *cNode* cible)	1024 octets toutes les 5 secondes

Ces simulations ont permis d'obtenir l'énergie résiduelle de chaque capteur à intervalles réguliers (une fois par minute). À partir de ces données, nous avons pu retracer les courbes d'évolution de la moyenne et de l'écart-type de la distribution de l'énergie résiduelle entre les nœuds. L'énergie résiduelle du cluster head a été volontairement ignorée. La valeur moyenne est présentée sur la figure 5.4. L'augmentation des valeurs à $t = 11$ minutes ainsi qu'à $t = 15$ minutes sur la courbe de la sélection par l'énergie provient de l'effet de récupération des batteries. Ces résultats sont conformes aux attentes : le processus de sélection par l'énergie consomme davantage d'énergie dans le réseau, ce que l'on peut imputer au rôle supplémentaire de *vNode* mis en place. Étant donné que les *vNodes* vont avoir à sortir périodiquement de leur état de veille pour envoyer une requête au *cNode* surveillé, écouter la réponse, et calculer une consommation théorique, le besoin en ressources énergétiques se retrouve logiquement affecté par rapport au processus de sélection pseudo-aléatoire.

Le volume de données de contrôle pour les deux méthodes proposées est donné par la figure 5.5. Ici encore, l'usage des *vNodes* entraine une augmentation des ressources

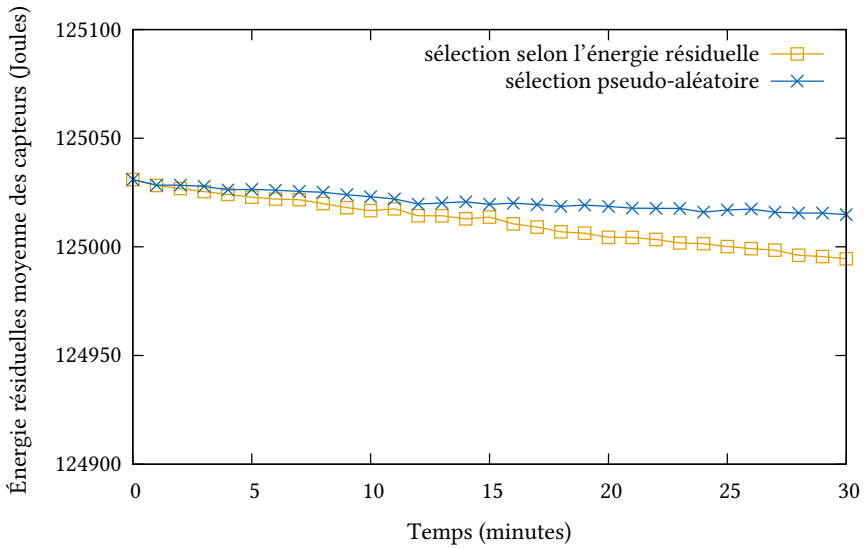

FIG. 5.4 : Valeur moyenne de l'énergie des nœuds (à l'exception du cluster head) en fonction du temps

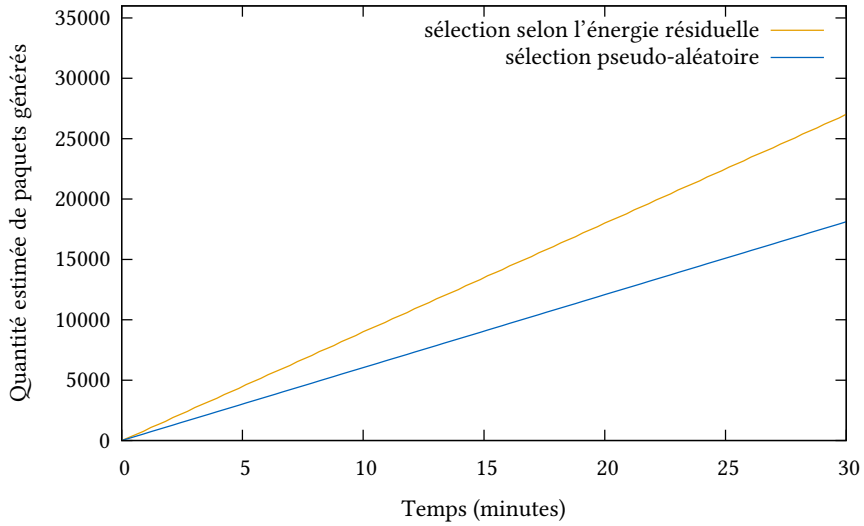

FIG. 5.5 : Estimation du nombre total de paquets générés au cours de la simulation au cours du temps

consommées ; il faut également tenir compte des messages (moins nombreux) envoyés par chaque capteur au moment du renouvellement de la sélection.

L'écart-type de la distribution en énergie résiduelle des nœuds est présenté sur la figure 5.6. Durant les premières minutes de la simulation, la sélection par l'énergie crée une dispersion plus affirmée de la charge en énergie du réseau, à cause des *vNodes* (il y a davantage de capteurs qui endossent des rôles consommant plus d'énergie). Mais après les sept premières minutes environ, l'écart-type pour le processus de sélection par l'énergie devient et demeure inférieur à celui associé à la méthode pseudo-aléatoire. Cela traduit la meilleure répartition de la charge énergétique dans le cluster, ce qui était l'objectif initial de la solution proposée dans ce chapitre. La différence entre les écart-types des deux méthodes est malgré tout peu élevée : cela tient entre autres aux modèles implémentés pour les simulations. Comme nous disposons d'un générateur de nombres pseudo-aléatoires efficace, à partir d'un certain nombre de renouvellements de la sélection, tous les capteurs vont endosser le rôle de *cNode* à peu près le même nombre de fois si la sélection se fait de façon pseudo-aléatoire. Et comme les capteurs ont également des activités de mesure identique, cette méthode (pseudo-aléatoire) fournit, dans notre cas, une excellente répartition de la charge dans le cluster. Dans une situation où les capteurs auraient des niveaux d'activité différents — si par exemple un évènement à détecter se produit beaucoup plus souvent dans une certaine région géographique du cluster — alors la consommation, avec cette méthode, ne serait plus nécessairement équilibrée entre tous les nœuds du cluster. Tandis que la sélection des *cNodes* par l'énergie résiduelle permettrait de rétablir cet équilibre.

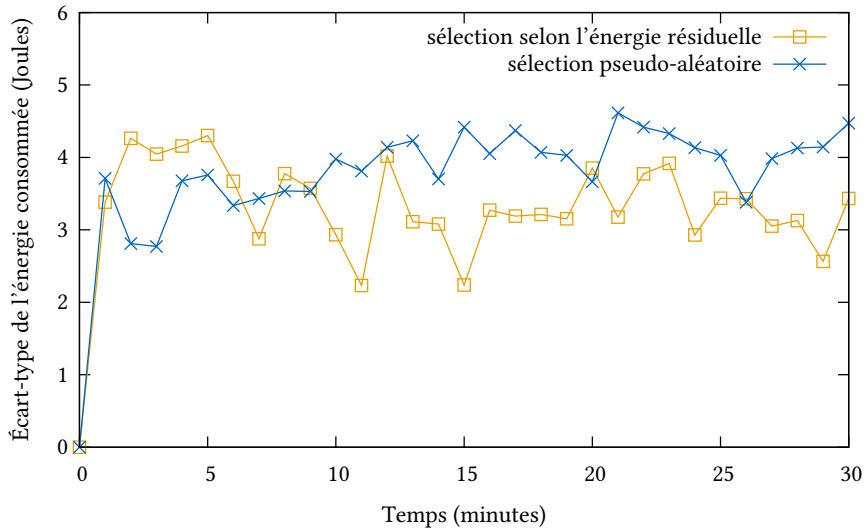

FIG. 5.6 : Écart-type pour l'énergie résiduelle des nœuds (à l'exception du cluster head) au cours du temps

2.2 Pousser plus loin l'amélioration

Par rapport au processus d'auto-désignation pseudo-aléatoire introduit dans le chapitre précédent, le second mécanisme proposé prend en compte l'énergie résiduelle de chaque nœud au moment de renouveler les *cNodes*, ce qui assure une meilleure répartition de la consommation en énergie dans le cluster. Mais si elle est mieux répartie, l'énergie consommée l'est aussi en plus grande quantité, principalement à cause de l'usage nouveau des *vNodes*. Les requêtes et les réponses aux *cNodes* consomment de l'énergie, et viennent alourdir l'implémentation de la solution : il y a trois rôles différents assignés dans le cluster (sans compter le cluster head), et deux d'entre eux nécessitent une étape spécifique pour leur attribution.

Il serait commode de pouvoir se passer des *vNodes*, mais sans faire de concessions sur la sécurité. L'idéal serait de pouvoir réutiliser les observations des *cNodes* antérieurs pour mettre en place un mécanisme de sélection qui tienne compte de l'énergie résiduelle, voire même d'un ensemble de paramètres permettant de choisir les meilleurs candidats. Ce jeu de paramètres, s'il inclue un score de confiance, peut même venir renforcer la sécurité du dispositif. C'est sur cet axe que s'appuient les améliorations proposées dans le prochain chapitre.

6

Élection démocratique des *cNodes*

Après avoir proposé un processus de sélection aléatoire, une première amélioration a été suggérée pour mieux répartir la consommation en énergie dans le réseau. Cette méthode basée sur le niveau d'énergie résiduelle des nœuds introduit de nouvelles considérations sur la sécurité du système, qui sont résolues par l'usage des *vNodes*. Mais ces derniers demandent un protocole de mise en place un peu plus complexe, et consomment de l'énergie ; dans ce chapitre, c'est une nouvelle méthode qui est proposée à la place.

Le processus de sélection reste basé sur l'énergie résiduelle des nœuds, mais toutes les mesures de vérifications sont affectées aux *cNodes*— puisqu'ils se chargent déjà de vérifier le trafic ! Il en résulte une méthode « démocratique », dans le sens où ce sont désormais les capteurs eux-mêmes qui « votent » auprès du cluster head pour déterminer les prochains *cNodes*. Encore une fois, des résultats numériques obtenus par simulation viennent éclairer les performances obtenues par ce mécanisme. Mis en perspective avec des instances basées sur les deux mécanismes précédents, ces résultats nous permettent d'établir une évaluation comparée des méthodes proposées, et de déterminer quel est le processus optimal à employer en fonction d'une situation donnée.

1 Processus d'élection démocratique des *cNodes*

Ce troisième processus de sélection est lui aussi proposé dans le contexte d'un renouvellement périodique des *cNodes*, et reprend les hypothèses présentées au chapitre 4, sous-section 1.1. Chaque renouvellement sera ici appelé une *itération* du processus. Pour exposer efficacement celui-ci, les notations suivantes sont adoptées :

- $\mathcal{N}$ désigne l'ensemble des capteurs du cluster (à l'exception du cluster head) ;
- $\mathcal{CN}$ désigne l'ensemble des *cNodes* du cluster à l'instant considéré ;
- i est un index utilisé pour parcourir $\mathcal{N}$;
- $\acute{E}R_k$ est un tableau contenant l'énergie résiduelle des nœuds annoncée par les *cNodes* au cluster head à la k^e itération.
- Obs_k est un tableau contenant les observations réalisées par le *cNode* j sur les communications de ses voisins.
- *LNÉ* est une liste dans laquelle seront placés les nœuds éligibles ; au contraire, *LNS* sera utilisée pour les nœuds suspects.
- *ÉCe* et *ÉCa* sont des variables contenant les valeurs d'énergie consommée effective et annoncée (respectivement) ; plus de détails sont fournis dans la suite.

Le processus d'élection débute, avant sa première itération, par une phase d'initialisation. Viennent ensuite les itérations elles-mêmes qui marquent le renouvellement de l'attribution du rôle de *cNode*, tant que le réseau est en fonctionnement.

1.1 Phase d'initialisation

Les étapes suivantes sont réalisées une fois le cluster formé :

- Chaque nœud i du cluster envoie au cluster head la valeur de son énergie résiduelle ; le CH l'enregistre dans un tableau spécifique, dénoté $\acute{E}R_0[i]$.
- Chaque nœud agit temporairement comme un *cNode*, et commence à surveiller le trafic de ses voisins. Il conserve en parallèle le fonctionnement d'un nœud normal (pour ce qui concerne les mesure physiques et la collecte de données), sans quoi il n'y aurait aucun trafic à observer. Cette étape a pour but d'essayer de repérer, dès le départ, d'éventuels nœuds compromis, pour les écarter de la liste des candidats au rôle de *cNode*.
- Une fois la durée de cette première étape d'observation écoulée, le processus d'élection démocratique peut commencer. La valeur du compteur d'itérations k est passée à 1.

1.2 Sélection des *cNodes*

Chaque itération k se décompose en étapes, qui sont détaillées ici :

1. La phase « active » de collecte de données a lieu. Les capteurs non sélectionnés amassent puis communiquent leurs données, sous la surveillance des *cNodes*. Ces derniers journalisent le nombre, la taille et la date des paquets envoyés par chacun de leur voisins, et signalent les éventuels comportements suspects au cluster head.

2. À la fin de l'itération k, chaque nœud i envoie son énergie résiduelle (dénotée $\acute{E}R_k[i]$) ; chaque *cNode* j envoie en plus un tableau $Obs_k[j]$ contenant les observations effectuées sur le débit d'émission de ses voisins au cours de l'itération.

3. Pour chaque nœud i, le cluster head réalise l'analyse suivante :

   - en se basant sur un modèle mathématique adapté, le CH calcule l'énergie consommée « effective » *ÉCe* relative à la moyenne des débits pour i $\{Obs_k[j][i]\}|_{j\in\mathcal{CN}}$ observés par les j *cNodes* voisins de i pendant le déroulement de l'itération [1] ;
   - le CH calcule alors la consommation « annoncée » *ÉCa* pour la période k, en prenant la différence entre les valeurs tout juste annoncées par i et celles de la période précédente : $\acute{E}Ca = \acute{E}R_{k-1}[i] - \acute{E}R_k[i]$;
   - si $|\acute{E}Ca - \acute{E}Ce| \leq \epsilon$ (où ϵ est la marge d'erreur tolérée), alors le nœud i est déclaré « sain », et il est ajouté à la liste *LNÉ* des nœuds éligibles pour la prochaine sélection des *cNodes* ;
   - sinon, le nœud est placé dans la liste *LNS* des nœuds suspects ;
   - soit $LNS[i]$ le nombre de fois où le nœud i a été déclaré suspect depuis le déploiement du réseau. Si $LNS[i] \geq$ seuil, alors le capteur est déclaré compromis et devient virtuellement exclu du cluster ; si à l'inverse le capteur avait systématiquement adopté un comportement honnête, il peut s'agir d'un faux positif, et le nœud n'est pas définitivement exclu. Il conserve la possibilité de rejoindre l'ensemble *LNÉ* au terme de prochaines itérations.

4. Une fois l'étape 3 terminée, le CH sélectionne les *cNodes* parmi l'ensemble *LNÉ* des nœuds éligibles. Le choix est réalisé de telle manière que tous les capteurs, *cNodes* compris, soient contrôlés par au moins deux *cNodes* distincts. Ainsi, un nœud compromis qui aurait néanmoins réussi à se faire élire en tant que *cNode* (avant de démarrer son attaque, par exemple), mais se comportant de façon malicieuse sur l'envoi de données, sera détecté.

5. Enfin l'itération suivante démarre : le compteur k est incrémenté, et l'on retourne à l'étape 1.

[1] La méthode proposée consiste ici à calculer la moyenne des valeurs relevées par les différents *cNodes* pour le trafic émis par le nœud i. Une autre solution serait de prendre la valeur maximale. Dans les deux cas, si malgré tout un capteur compromis est parvenu à accéder au rôle de *cNode*, il pourrait tenter de fournir des valeurs anormalement élevées pour fausser le calcul de *ÉCe* et empêcher l'élection du nœud i. Une solution idéale serait en fait de calculer une moyenne pondérée, en utilisant comme coefficients les degrés de confiance associés à chaque *cNode*.

1.3 Modèle mathématique pour la consommation en énergie

Tout comme dans le chapitre précédent, la solution présentée ici implique des estimations sur la consommation en énergie des capteurs du cluster. Nous suggérons à nouveau pour ces calculs d'utiliser le modèle de diffusion proposé par Rakhmatov et Vrudhula, détaillé au chapitre 5, sous-section 1.1.

1.4 Sélection des *cNodes* parmi l'ensemble *LNÉ*

À l'étape 3 du processus d'élection, les cluster heads choisissent les *cNodes* pour l'itération à venir parmi tous les nœuds placés dans la liste *LNÉ*. Les critères retenus pour effectuer ce choix peuvent être très variés. Par exemple, il est possible de tirer aléatoirement des *cNodes* parmi les nœuds de *LNÉ*, jusqu'à ce que :

- l'on ait assez de *cNodes* (selon le seuil déterminé par l'utilisateur du réseau) ;
- et que tous les nœuds du cluster soient couverts par au moins deux *cNodes*, comme expliqué précédemment.

D'autres façons d'effectuer ce choix pourraient consister par exemple à prendre en compte un ou plusieurs critères parmi la liste suivante :

- l'énergie résiduelle des nœuds ;
- la confiance attribuée à ce nœud (sous la forme d'un score de réputation) ;
- l'index de connectivité (le nombre de voisins directs des nœuds) ;
- la puissance du signal ;
- *et cætera.*

Plusieurs de ces critères peuvent être pondérés et combinés de façon à obtenir une note pour chaque nœud, comme par exemple dans l'équation qui suit :

$$note_k[i] = (\alpha \times \acute{E}R_k[i]) + (\beta \times r\acute{e}p_k[i]) + (\gamma \times ic_k[i]) + (\delta \times ps_k[i]) + (\zeta \times ndd_k[i])$$

où :

- $note_k[i]$ représente la note du nœud i pour l'itération k ;
- $\acute{E}R_k[i]$ reste l'énergie résiduelle pour le même nœud et la même itération ;
- $r\acute{e}p_k[i]$ est le score de réputation du nœud i (mis à jour au fur et à mesure, selon les signalements ou les scores envoyés par les *cNodes* à propos de i) ;
- $ic_k[i]$ est l'index de connectivité de i ;
- $ps_k[i]$ est la puissance moyenne du signal enregistrée par les voisins du nœud i ;
- $ndd_k[i]$ est la valeur maximale entre un seuil prédéterminé et le nombre d'itérations antérieures à k depuis que le nœud a été désigné *cNode* pour la dernière fois ;
- α, β, γ, δ et ζ sont des constantes définies par l'utilisateur.

Cette formule pourrait être utilisée pour classer les nœuds de la liste *LNÉ* de telle manière que le cluster head puisse choisir les meilleurs *cNodes* possibles par rapport aux critères pris en compte.

La détermination des critères à retenir et des poids à appliquer revient à l'exploitant du réseau ; en pratique, il faudrait les adapter par rapport à l'application qui sera faite du réseau de capteurs, tout en tenant compte des conditions particulières de l'environnement dans lequel celui-ci sera déployé. Dans un réseau constitué de nœuds statiques, par exemple, l'index de connectivité des capteurs, de même que la puissance de leur signal, ne sont pas censés subir de modifications sensibles entre deux itérations consécutives. En conséquence, les poids de ces deux critères devraient être modérés (par rapport aux autres poids utilisés dans la formule) afin d'éviter de désigner les mêmes capteurs à chaque itération. Dans un réseau clusterisé, où tous les nœuds peuvent atteindre leur CH en un seul saut, l'index de connectivité ou la puissance du signal ne constituent peut-être pas des critères pertinents pour la formule de sélection (bien que, là encore, cela puisse dépendre de l'application déployée). Mais même si nous travaillons principalement dans des réseaux clusterisés dans cet ouvrage, il est bon de se rappeler que tous les réseaux de capteurs ne font pas nécessairement appel à de tels mécanismes. Et que nombre d'entre eux comportent également des nœuds mobiles. Dans de tels environnements, l'évaluation de la densité des nœuds ou de la qualité des liens dans les zones où sont situés les *cNodes* candidats devient plus intéressante, car il s'agit souvent d'indicateurs précieux sur les performances du réseau.

Toujours sur le même propos, la contrainte fixée consistant à faire en sorte que chaque nœud soit surveillé par au moins deux *cNodes* peut éventuellement être aménagée. S'il s'agit en principe d'une bonne assurance pour détecter les nœuds compromis dans des clusters où tous les nœuds sont réunis à portée d'émission de leur cluster head, cette contrainte pourrait s'avérer trop lourde pour certaines topologies. Ainsi dans un réseau en étoile, par exemple, où tous les capteurs se retrouveraient sur des branches et n'auraient que deux voisins (l'un plus près, l'autre plus loin de la station de base), la mise en application de la contrainte pourrait avoir pour effet de désigner tous les nœuds du réseau comme *cNodes*.

Quelques remarques sur le processus dans son ensemble : par rapport au mécanisme proposé dans le chapitre précédent (sélection selon l'énergie résiduelle : chaque nœud annonce son énergie, et « vote pour lui-même » en quelque sorte), le processus présenté ici confie la vérification de la consommation en énergie aux *cNodes* et non plus à des *vNodes* expressément nommés dans ce but. Les *cNodes* vont donc envoyer au cluster head, au moment du renouvellement de la sélection, un compte-rendu des observations réalisées pour chaque nœud : estimation de l'énergie consommée, mais aussi, suivant les détails du modèle implémenté : score de réputation (basé sur le respect ou non des règles), puissance du signal reçu, et autres critères comme évoqué plus haut. C'est ce vote « pour les autres » qui fait de la méthode un processus « démocratique » : en envoyant leur compte-rendu au CH les *cNodes* votent en quelque sorte pour déterminer leurs successeurs. Et par « successeurs », nous touchons d'ailleurs un aspect important : si la sélection est basée sur d'autres critères que l'énergie résiduelle [2], il est recommandé de faire en sorte que les *cNodes* ne puissent pas « voter » pour eux-mêmes. De cette manière, si un nœud compromis accède au rôle, il ne sera pas en mesure de le conserver ; à moins de s'être comporté de façon irréprochable et

[2] Dans le cas où seule l'énergie résiduelle est prise en compte, le surcroit de consommation induit par la surveillance devrait suffire à faire baisser le niveau d'énergie du nœud en dessous de celui des autres capteurs, et à assurer ainsi la rotation.

d'être sollicité par des *cNodes* voisins (ce qui signifierait qu'il ne met pas à mal le bon fonctionnement du réseau).

Une autre conséquence de ce vote démocratique, suivant l'implémentation réalisée pour le choix des *cNodes*, est que les nœuds possédant peu de voisins peuvent être amenés à devenir moins souvent *cNodes* (s'ils ont moins de *cNodes* voisins qui votent pour eux, et que les votes ne sont pas pondérés par le cluster head en fonction du nombre reçu). Mais d'un autre côté, s'ils ont peu de voisins, il n'est sans doute pas très utile de les sélectionner (à la fois parce qu'ils n'auraient que peu de voisins à surveiller, et parce que si la zone géographique où ils sont situés est peu couverte par les capteurs, mieux vaut qu'ils continuent à réaliser leur opération de captage).

2 Comparaison numérique des méthodes de sélection

2.1 Mise en place des simulations

Le modèle utilisé dans cette section est similaire à celui employé au chapitre 4 (défini en sous-section 2.1). Tout comme alors, le logiciel ns a été mis en œuvre. Le cluster est composé d'une grille carrée de cent capteurs, avec le cluster head en son centre (soit cent-un capteurs au total ; voir figure 4.6).

La table 6.1 récapitule les principaux paramètres impliqués dans l'exécution des instances.

Tab. 6.1 : Paramètres de simulation

Paramètre	Valeur
Durée de la simulation	3 600 secondes
Nombre de capteurs	100 (+ cluster head)
Nombre de *cNodes*	7–10 (selon scénario)
Nombre de nœuds compromis	1
Fréquence de renouvellement des *cNodes*	toutes les 10 secondes
Durée de la période initiale pour la sélection démocratique	60 secondes
Mobilité des nœuds	nulle

Paramètre	Valeur (nœuds normaux)	Valeur (nœud compromis)
Taux d'émission	1 ko/s	35 ko/s
Taille des paquets	500 octets	100 octets
Intervalle	aléatoire (Poisson)	constant
Cons. énergétique en émission	0,660 W	0,660 W
Cons. énergétique en réception	0,395 W	non prise en compte
Quantité initiale d'énergie	10 J — ∞ (sel. sc.)	∞

Dans les chapitres précédents, le but des simulations était de souligner les apports des nouvelles solutions proposées par rapport aux éléments antérieurs. Ici, l'objectif consiste plutôt à comparer tous les scénarios que nous avons définis au cours des trois chapitres sur les processus de sélection. Un « scénario » se caractérise donc

avant tout par le choix d'un processus de renouvellement des *cNodes*, mais aussi par le nombre de *cNodes* sélectionnés à chaque tour, et par le montant d'énergie initialement attribué aux capteurs. Chaque graphe présente, pour un montant d'énergie initiale donné, les valeurs obtenues pour cinq scénarios distincts, auxquels nous attribuons des identifiants pour pouvoir nous y référer plus facilement par la suite :

- l'un ne comporte aucun renouvellement des dix *cNodes* (il s'agit de la méthode dite « statique », que nous nommons dans cette section #Stat10) ;
- l'un se base sur un processus de renouvellement aléatoire tel que présenté au chapitre 4 (#Aléa10) ;
- l'un se base sur un renouvellement selon l'énergie résiduelle (avec usage de *vNodes*) tel que présenté au chapitre 5 (#ÉRes10) ;
- les deux derniers se basent sur le mécanisme d'élection démocratique abordé dans ce chapitre :
  - l'un avec dix *cNodes* (#Démc10),
  - l'autre avec sept *cNodes* seulement, afin d'observer l'effet de la diminution du nombre de nœuds sélectionnés (#Démc07).

Sauf indication contraire, tous les résultats numériques présentés dans cette section sont des moyennes calculées à partir des valeurs de dix instances distinctes (pour chaque scénario). Ces instances sont différenciées par les valeurs utilisées pour initialiser le générateur de nombres pseudo-aléatoires de ns, et (le cas échéant) celui utilisé par les nœuds pour la sélection des *cNodes*.

Les moyennes sur les capteurs sont calculées à partir des valeurs des quatre-vingt-dix-neuf nœuds « sains ». Le cluster head ainsi que le nœud compromis sont dotés d'une réserve d'énergie illimitée pour les simulations :

- le cluster head parce qu'il est indispensable au fonctionnement du cluster, et que sur une instance réelle, l'épuisement de sa batterie se traduirait par la désignation immédiate d'un remplaçant ; de plus, il reçoit les paquets « utiles » de tous les nœuds, et sa consommation en énergie, très importante, viendrait « fausser » les moyennes que nous cherchons à étudier ici ;
- le nœud compromis parce que nous ne souhaitons pas qu'il arrête d'émettre au cours du déroulement de notre scénario ; et parce que les nombreux paquets qu'il émet lui font là aussi dépenser une énergie qui viendrait fausser les moyennes observées. Outre le contexte des simulations, il est à noter qu'un tel cas est plausible, par exemple, si l'attaquant substitue au nœud ciblé un appareil plus puissant (un ordinateur portable par exemple) sur une longue durée : le nœud compromis peut alors se retrouver affranchi des contraintes en énergie auxquelles sont soumis les autres capteurs.

2.2 Résultats numériques

2.2.1 ▶ Consommation énergétique

Une première série d'instances a été exécutée en attribuant une quantité d'énergie « illimitée » aux capteurs au regard de la durée virtuelle de la simulation. Ceci permet d'observer la quantité d'énergie moyenne consommée par les nœuds lorsque chaque méthode de sélection des *cNodes* est déployée dans le cluster. La figure 6.1 présente les valeurs obtenues. On peut y observer une consommation quasiment identique entre les méthodes #Stat10 et #Aléa10, puisque le renouvellement en soi, à l'aide de la sélection pseudo-aléatoire des *cNodes*, ne représente pratiquement aucun cout additionnel. Les méthodes #ÉRes10 et #Démc10 consomment toutes deux davantage, à cause des mécanismes mis en place pour récolter les valeurs de l'énergie résiduels des nœuds, et pour assurer la sécurité du processus.

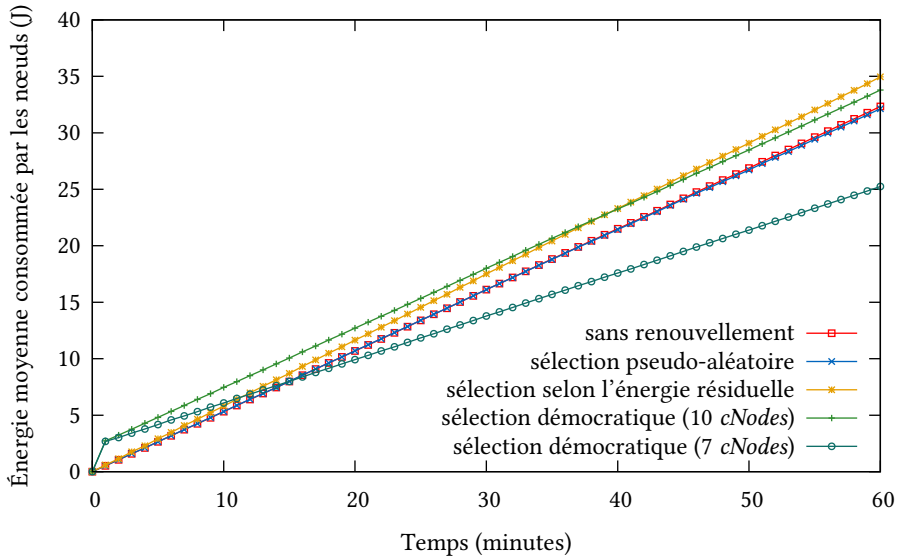

FIG. 6.1 : Moyenne de l'énergie consommée par les capteurs au cours du temps (énergie initiale : ∞)

La méthode #Démc10 est plus couteuse au départ, ce qui est parfaitement visible en agrandissant la vue sur l'origine du repère comme le fait la figure 6.2. Les deux scénarios basés sur la sélection démocratique voient leur quantité d'énergie consommée augmenter très rapidement pendant la première minute de la simulation, qui correspond à la période initiale (où tous les nœuds agissent comme des *cNodes* en plus de prendre des mesures, d'où la consommation accrue). En revanche, on observe une inversion entre #ÉRes10 et #Démc10 peu avant 40 minutes, signe que sur le long terme, #Démc10 consomme moins d'énergie que l'usage des *vNodes*. #Démc07, quant à elle, est à terme la méthode la plus économe en énergie : ceci met en avant le poids des *cNodes* dans la consommation totale en énergie du cluster. Moins on utilise de *cNodes*, moins l'énergie consommée est importante (comme indiqué déjà par les résultats du chapitre 4, en section 2).

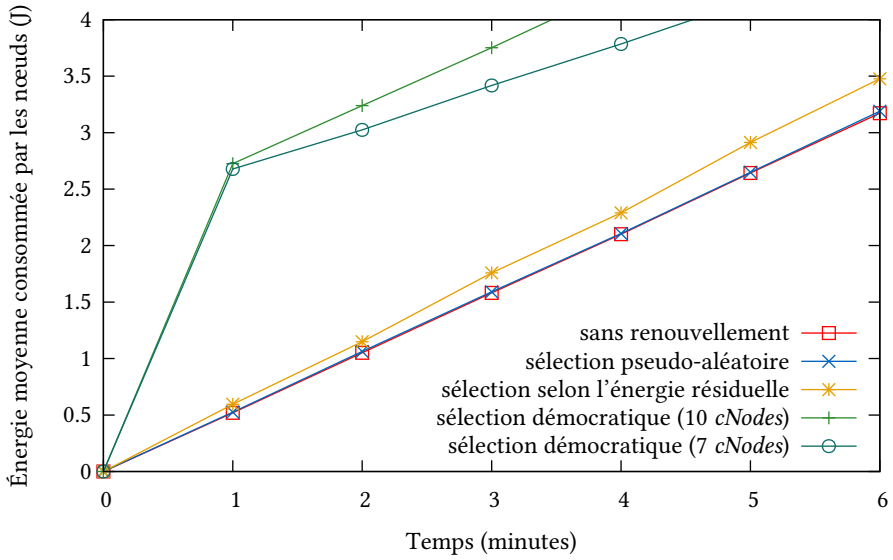

FIG. 6.2 : Moyenne de l'énergie consommée par les capteurs au cours du temps : agrandissement de la vue près de l'origine du repère (énergie initiale : ∞)

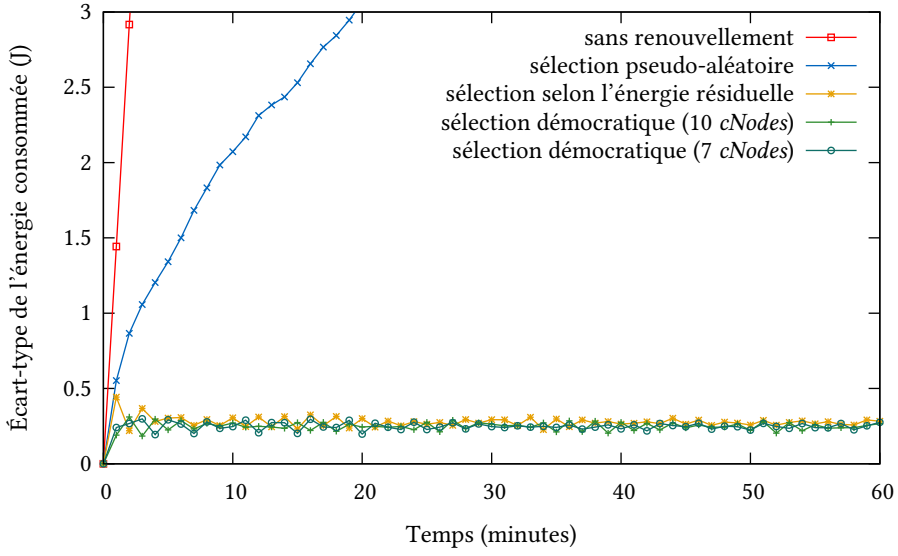

FIG. 6.3 : Écart-type moyen pour l'énergie consommée par les capteurs au cours du temps (énergie initiale : ∞)

Les méthodes qui prennent en compte l'énergie résiduelle (#ÉRes10, #Démc10) sont donc, pour un nombre de *cNodes* équivalent, plus gourmandes en énergie ; mais il est aussi intéressant d'étudier la répartition de la charge qu'elles apportent au cluster. La figure 6.3 présente les écarts-types obtenus, au court des mêmes simulations, pour les cinq scénarios. On peut y observer que pour #Stat10 se crée un déséquilibre croissant (de façon linéaire) : puisque les *cNodes* sont toujours les mêmes, ce sont toujours les mêmes capteurs qui consomment l'énergie, creusent l'écart, et diminuent l'équilibre du réseau. Pour la méthode #Aléa10, l'écart monte rapidement, mais ralentit et se stabilise peu à peu (la valeur (toujours croissante) en fin de simulation, non apparente sur la figure, est à un peu plus de 6 J) : la loi des grands nombres fait que (sous réserve de l'utilisation d'un générateur de nombres pseudo-aléatoires bien conçu), lorsque la durée de simulation s'allonge, tous les nœuds sont sélectionnés un nombre de fois approximativement identique. Mais cet écart demeure très important à côté de celui de #ÉRes10, #Démc10 et #Démc07, très proche de zéro : ces méthodes assurent une très bonne répartition de la consommation en énergie dans le cluster, ce qui, pour rappel, constituait leur objectif initial.

2.2.2 ▶ Durée de vie du réseau

Une deuxième et une troisième série d'instances ont été réalisées pour analyser la durée de vie du réseau ainsi que le taux de détection de l'attaque. C'est d'abord la durée de vie des capteurs qui est étudiée ici. Son évolution a été observée au cours du temps, dans un cluster doté d'une énergie initiale de 10 J (deuxième série) puis de 20 J (troisième série) par nœuds. Ce montant d'énergie est suffisamment bas pour observer l'arrêt de la plupart des nœuds au cours de l'heure virtuelle simulée.

La figure 6.4 présente ainsi le nombre de nœuds encore en activité en fonction du temps, lorsqu'une énergie de 10 J a initialement été affectée aux capteurs. La méthode

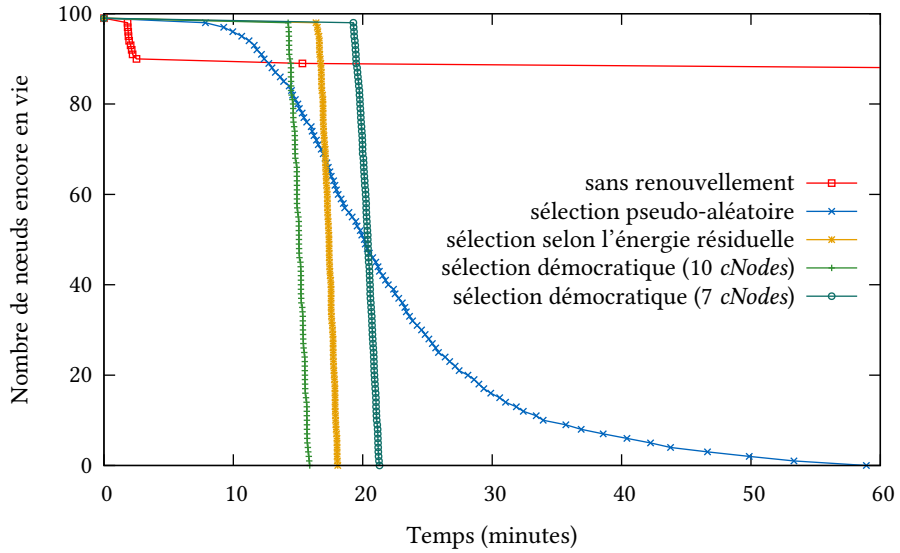

FIG. 6.4 : Nombre de nœuds en activité au cours du temps pour un montant d'énergie initiale de 10 J

#Stat10 est de loin celle qui conserve ses nœuds le plus longtemps. Et pour cause : comme les *cNodes* ne sont pas renouvelés, les 10 capteurs initialement sélectionnés sont les seuls à consommer de manière accrue, et à vider (très rapidement, d'ailleurs) leur batterie. Le dixième nœud à épuiser sa batterie s'arrête longtemps après le neuvième : cela est dû au calcul des moyennes sur plusieurs instances. Il arrive en effet que le nœud compromis soit sélectionné parmi les *cNodes* ; mais comme sont énergie n'apparait pas sur la courbe, les résultats sont ceux obtenus pour seulement 9 *cNodes* sélectionnés dans ce cas, et le dixième nœud à tomber hors service est un nœud normal qui survit largement à l'heure que dure la simulation.

La méthode de renouvellement aléatoire #Aléa10 est efficace pour préserver l'énergie dans le cluster. Comme le nombre de *cNodes* sélectionnés est statistiquement un pourcentage du nombre de nœuds dans le cluster (on sélectionne k nœuds de façon aléatoire, sans regarder s'ils sont encore en vie), la réduction du nombre de nœuds n'entraine pas l'accélération de la consommation. De plus, la méthode est simple à mettre en œuvre, et ne nécessite que peu d'échanges de données de contrôle.

Les méthodes #ÉRes10, #Démc10 et #Démc07 en revanche ne sont pas aussi efficaces dans ce contexte. L'énergie est consommée plus rapidement que ce soit à cause de la période initiale (#Démc10 et #Démc07) ou bien par le recours aux *vNodes*. Il en résulte un arrêt plus rapide qu'avec #Aléa10. Et surtout, ce premier arrêt est très rapidement suivi par celui des autres capteurs, ceci pour les deux raisons suivantes :

- l'écart-type de l'énergie consommée étant très faible dans le cluster, tous les capteurs ont à peu près la même consommation, et leur énergie résiduelle atteint un niveau nul au même moment ;

- le fait de sélectionner les 10 *cNodes* parmi les nœuds ayant le plus d'énergie résiduelle provoque la désignation systématique de 10 *cNodes*, puisque l'on ne risque pas de désigner « par erreur » des nœuds dont la batterie seraient épuisée. Lorsque les premiers nœuds s'arrêtent, le pourcentage de *cNodes* par rapport aux capteurs non sélectionnés augmente donc mécaniquement, contrairement à la méthode #Aléa10, et pousse les nœuds restant à vider rapidement leur batterie.

On peut observer que de ces méthodes basées sur l'énergie résiduelle, #Démc07 est la plus efficace ici (puisqu'elle met en jeu moins de *cNodes*), tandis que #ÉRes10 se classe deuxième et #Démc10 dernière, pénalisée par l'énergie consommée au cours de sa période initiale. Pourtant l'écart creusé par cette période initiale peut être rattrapé. C'est ce que montre la figure 6.5, qui reprend les mêmes scénarios, mais développés sur la consommation des 20 J (au lieu de 10 J) initialement attribués au capteurs : #Démc10 et #ÉRes10 sont quasiment à égalité dans ce cas. On retrouve un comportement à peu près identique pour #Stat10 et #Aléa10 par rapport à la figure 6.4, mais bien évidemment avec une durée d'épuisement des nœuds doublée. Tout comme #Démc10 grignote du terrain sur #ÉRes10, on observe que #Démc07 gagne en efficacité en comparaison de #Aléa10 : avec 10 J d'énergie initiale, les nœuds s'épuisent d'un coup pour #Démc07 peu après 20 minutes, alors qu'il reste près de la moitié des nœuds pour la méthode #Aléa10 au même instant. Tandis qu'avec 20 J d'énergie initiale, l'écart créé par la période initiale de #Démc07 a été en partie rattrapé, et lorsque le nombre de nœuds chute peu avant les 50 minutes de simulation, près de trois quarts des capteurs sont épuisés avec #Aléa10. Avec une quantité plus élevée d'énergie initiale, le nombre inférieur de *cNodes* ferait rapidement la différence en faveur de la méthode #Démc07.

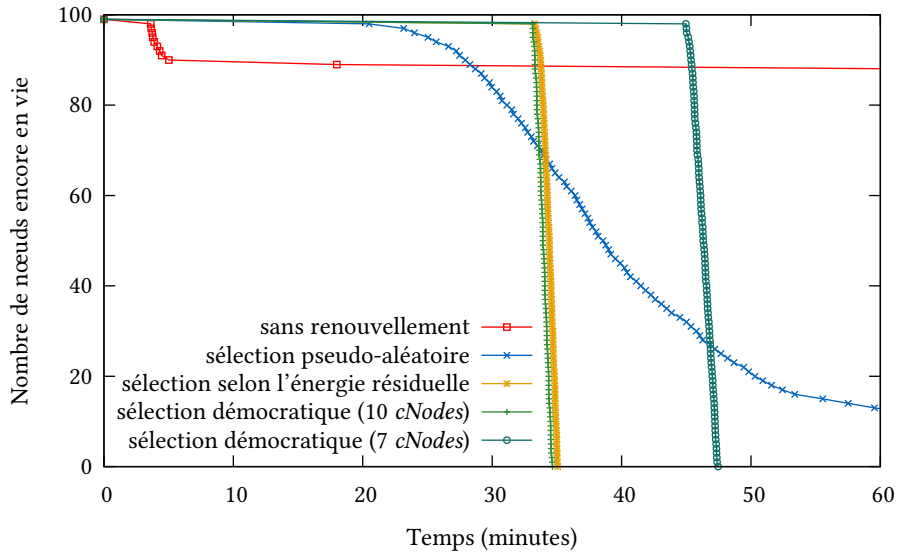

FIG. 6.5 : Nombre de nœuds en activité au cours du temps pour un montant d'énergie initiale de 20 J

2.2.3 ▶ Taux de détection

Le nombre de nœuds en mesure de détecter l'attaque du nœud compromis est directement lié au nombre de nœuds encore en activité dans le réseau, puisque les capteurs ayant épuisé leur batterie n'ont plus aucune chance de mener à bien l'opération de surveillance. À titre indicatif, la figure 6.6 représente le taux de détection à chaque phase (de 10 secondes), sur une seule instance (pas de moyenne, donc), pour chaque méthode de sélection, l'énergie initiale assignée aux capteurs étant de 10 J. En dehors d'un intérêt artistique certain, ces courbes permettent d'observer les extremums atteints, mais sont très peu lisibles. La figure 6.7 reprend donc ces valeurs, lissées une première fois en calculant la moyenne du nombre d'attaques détectées par les différentes instances (10 par scénario), puis une seconde fois sur une séquence d'une minute (donc 6 phases de 10 secondes). On observe que pour la méthode #Stat10, la détection chute très rapidement à zéro ; en fait dès que les *cNodes* d'origine, non renouvelés, ont consommé leurs réserves. Les méthodes #Aléa10, #ÉRes10 et #Démc10 ont toutes les trois un bon niveau de détection, avec en moyenne 5 *cNodes* parmi les 10 sélectionnés qui repèrent l'attaque tant que la plupart des nœuds sont en activité. La méthode #Démc07 tourne plutôt sur une moyenne de 3 à 4 *cNodes* détectant l'attaque, mais sur un total de 7. La détection se poursuit en revanche un peu plus longtemps que sur #ÉRes10 et #Démc10, puisque les nœuds consomment au total moins d'énergie. Les maigres réservent accordées ne lui permettent pas d'avoir le temps, en revanche, de passer devant la sélection aléatoire #Aléa10, qui détecte encore parfois l'attaque au-delà de 30 minutes de simulation.

Il est à noter que la position du nœud compromis a une influence directe sur le nombre de nœuds détectant l'attaque. Suivant le nombre de voisins directs qu'il possède, ce nœud a plus ou moins de chances d'être détecté. Pour toutes les simulations réalisées dans cette partie, le nœud compromis occupe la position indiquée sur la figure 6.8, et possède (en dehors du cluster head) un total de 61 voisins directs, qui

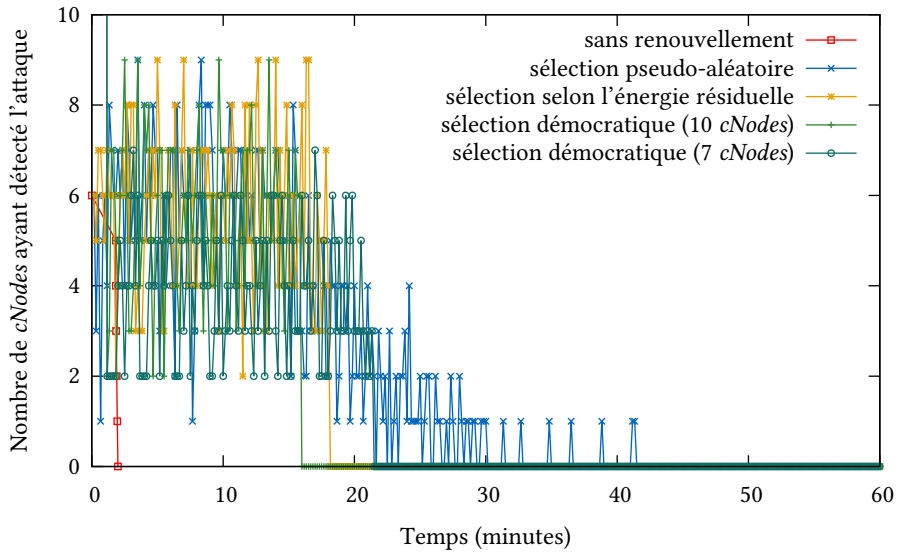

FIG. 6.6 : Taux de détection, pour chaque processus de sélection, au cours du temps (énergie initiale : 10 J ; valeurs sur une seule instance)

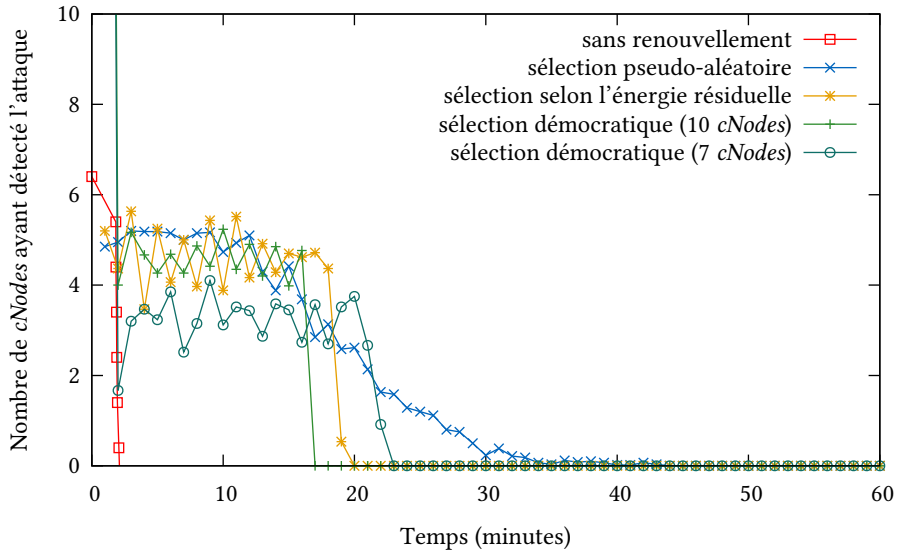

FIG. 6.7 : Taux de détection, pour chaque processus de sélection, au cours du temps (énergie initiale : 10 J ; valeurs lissées sur une durée d'une minute)

sont autant de *cNodes* en puissance capables de détecter l'attaque [3]. Toutefois, « surveillance » n'est pas systématiquement égale à « détection », et un *cNode* à portée peut échouer à détecter l'attaque (s'il manque trop de paquets à cause des collisions, par exemple).

Cette remarque sur la disposition permet de réaliser une seconde observation sur le nombre de chances que possède le nœud compromis d'échapper, à propos de la couverture géographique, à la surveillance des *cNodes*. Avec les méthodes #ÉRes10, #Démc10 et #Démc07, cette chance est nulle (tant qu'il reste suffisamment de nœuds actifs), puisque les méthodes employées spécifient que chaque nœud du cluster doit être couvert par au moins deux *cNodes*. Mais le processus #Aléa10, en revanche, est tel que pour certaines phases, tous les *cNodes* peuvent être sélectionnés parmi l'ensemble des nœuds non voisins du nœud compromis. Cette probabilité se calcule. La répartition des *cNodes*, lors du processus de sélection aléatoire, suit une loi de probabilité hypergéométrique. Il est possible de calculer la probabilité pour que k *cNodes* sur les $n = 10$ soient choisis parmi les $m = 61$ voisins directs du nœud compromis, sur un total de $N = 100$ candidats, à l'aide de la formule suivante :

$$P(k) = \frac{\binom{m}{k}\binom{N-m}{n-k}}{\binom{N}{n}} = \frac{m!}{k!\,(m-k)!} \times \frac{(N-m)!}{(n-k)!\,((N-m)-(n-k))!} \times \frac{n!\,(N-n)!}{N!}$$

La probabilité que le nœud compromis ne soit surveillé par aucun *cNode* est donc de :

$$P(k=0) \approx 3,6 \times 10^{-5} \approx 1/27228$$

Elle est très faible ; mais il faut tenir compte encore une fois du fait que la surveillance peut échouer et ne pas mener à la détection de l'attaque, que ce soit en raison d'un nombre important de collisions dans le cluster qui viendrait perturber l'écoute du trafic, ou parce que sur une période donnée le capteur compromis a émis légèrement moins de paquets que d'habitude et que le seuil n'est pas atteint pour toutes les sentinelles.

La figure 6.9 présente les mêmes résultats, pour une énergie initiale de 20 J assignée aux capteurs. Les résultats, de manière prévisible, montrent que la durée de surveillance a à peu près doublé pour chaque méthode, avec une augmentation des performances pour #Démc10 et #Démc07 (par rapport à la figure précédente), qui ont eu l'occasion de « rattraper » davantage l'écart creusé sur la consommation en énergie lors de leur période initiale. Avec encore un peu plus d'énergie allouée au début de la simulation, #Démc10 permettrait au capteurs de maintenir la détection plus longtemps que #ÉRes10, et à terme tendrait vers une efficacité presque équivalente à #Aléa10, sans la rattraper toutefois en raison d'un volume de données de contrôle légèrement supérieur.

Au vu des caractéristiques obtenues pour chaque méthode, nous pouvons à présent résumer leurs avantages et leurs inconvénients, et en déduire des recommandations d'usage selon la configuration du réseau déployé.

[3] Le modèle de couche physique utilisé pour les simulations ne comporte pas de composante aléatoire : hors collisions, soit deux nœuds sont à portée et reçoivent tous leurs messages sans pertes dues à l'affaiblissement, soit l'échange est systématiquement impossible.

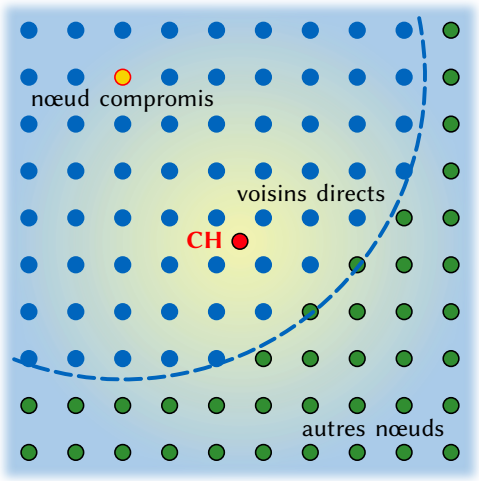

FIG. 6.8 : Schéma du cluster représentant la position des soixante-et-un voisins directs (en bleu) du nœud compromis (en jaune) tels que définis dans les simulations réalisées

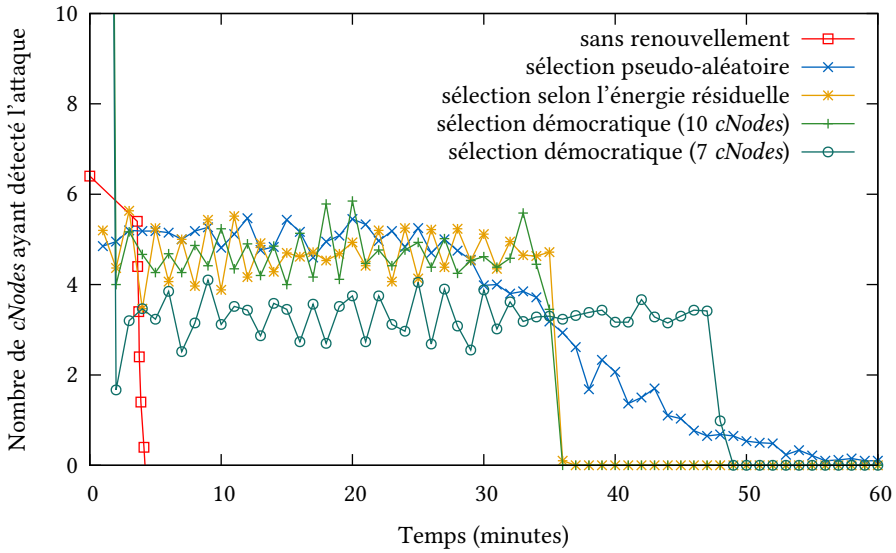

FIG. 6.9 : Taux de détection, pour chaque processus de sélection, au cours du temps (énergie initiale : 20 J ; valeurs lissées sur une durée d'une minute)

3 Choisir un processus de sélection

3.1 Choisir un processus adapté à la situation

Plusieurs méthodes de sélection des *cNodes* ont été proposées, et leurs performances ont été mesurées à travers un jeu de simulations. Il est temps désormais d'en tirer des conclusions, et de déterminer quelle méthode est préférable pour un cas d'utilisation donné. Car il n'y a pas vraiment de « meilleure méthode » adaptable à toutes les applications, chacune possède ses avantages et ses inconvénients.

Aucun renouvellement des *cNodes* Ne pas renouveler les *cNodes* permet de garder en vie très longtemps les nœuds qui ne sont pas sélectionnés, mais n'assure que très mal la sécurité du réseau. Les *cNodes* d'origine vont rapidement épuiser leur batterie et tomber hors service. De plus, si la sélection est réalisée de telle sorte que tous les capteurs du cluster ne soient pas couverts par un *cNode* [4], alors certains nœuds ne seront jamais surveillés. La mise en place des *cNodes* sans renouvellement n'est donc pas conseillée, en dehors de cas particuliers où les *cNodes* sont connus d'avance, disposent de composants matériels distincts des autres capteurs (une meilleure batterie, notamment), et sont expressément disposés de manière à assurer une couverture suffisante du cluster.

Renouvellement périodique : sélection pseudo-aléatoire Le processus de renouvellement des *cNodes* par sélection aléatoire est un excellent compromis entre sécurité et longévité du réseau : cette méthode, simple à mettre en place, ne requiert que peu d'échanges de données de contrôle, et consomme peu d'énergie (en dehors du surcout inévitable induit par l'usage des *cNodes*). Lorsque les premiers nœuds épuisent leur batterie, le nombre moyen de *cNodes* élus s'adapte, puisqu'il s'agit d'un pourcentage souhaité sur les nœuds en activité, et non d'un nombre prédéfini fixe. L'équilibre en matière de consommation énergétique ne sera pas aussi bon qu'avec d'autres méthodes : l'application ne doit pas craindre de perdre certains capteurs avant les autres. De même, le niveau de sécurité n'est pas à son maximum, puisque certains capteurs peuvent se retrouver en dehors du champ de couverture des *cNodes* pour une phase donnée. Cependant, le renouvellement fréquent et les lois des probabilités font qu'un nœud compromis devrait être rapidement détecté, au plus au bout de quelques phases successives.

Renouvellement périodique : sélection selon l'énergie résiduelle L'utilisation des *vNodes* permet de sécuriser le processus de renouvellement selon l'énergie résiduelle, mais au prix d'un cout très important en énergie. La répartition de la charge sur les capteurs du cluster est excellente, mais ne dépasse pas celle obtenue par la méthode d'élection démocratique, qui consomme moins d'une manière générale, une fois sa phase initiale amortie. Le seul cas d'usage pour cette méthode serait éventuellement un réseau pour lequel l'équilibre en énergie résiduelle des capteurs est important, mais

[4] Ou même, en réalité, par le nombre minimal de *cNodes* nécessaires pour reporter effectivement une attaque, puisque par principe le cluster head attend de recevoir des alarmes de la part de plusieurs *cNodes* distincts avant de déclarer un capteur comme compromis, ceci afin d'éviter les faux positifs (accidentels ou d'origine malveillante).

qui ne connaitrait que de faibles périodes d'activités, par exemple si tous les capteurs s'allument sur une durée de quelques minutes, effectuent des mesures, puis s'éteignent pour plusieurs heures. Dans les autres cas, le processus d'élection démocratique lui sera préféré.

Renouvellement périodique : élection démocratique Les performances du processus d'élection démocratique sont similaires à celles de la méthodes selon l'énergie résiduelle pour ce qui est de l'équilibre de la charge dans le cluster, c'est-à-dire qu'elles sont excellentes. La consommation générale en énergie est meilleure, une fois la période initiale « amortie », mais elle est loin d'égaler celle de la méthode de sélection aléatoire. La contrainte imposée de couvrir chaque capteur par au moins deux *cNodes* assure un bon niveau de surveillance, et la sécurité du dispositif est garantie par les « votes » des *cNodes* qui aident le cluster head à désigner leurs successeurs. Cette méthode possède un autre avantage : en réutilisant la surveillance effectuée par les *cNodes* lorsque vient le moment de renouveler la sélection, il est possible d'utiliser d'autres facteurs pour choisir les nœuds, comme la puissance des signaux émis, l'index de connectivité des capteurs, ou encore leur indice de confiance. Cette méthode est utilisable pour des applications qui ont des contraintes fortes en termes d'équilibre de la consommation, c'est-à-dire des applications qui doivent conserver l'ensemble des capteurs en fonctionnement le plus longtemps possible.

Les avantages et inconvénients de chaque méthode sont résumés en table 6.2. La table 6.3, quant à elle, synthétise les cas d'utilisation de ces processus.

Tab. 6.2 : Avantages et inconvénients des différents processus de sélection

Méthode	Avantages	Inconvénients
Aucun renouvellement	• Préservation des nœuds non sélectionnés	• Mauvaise surveillance
Sélection aléatoire	• Consommation d'énergie modérée • Simple à mettre en œuvre • Le nombre de *cNodes* sélectionnés est statistiquement un pourcentage constant sur le nombre de capteurs en activité • Bonne rotation des *cNodes* ; processus aléatoire, donc pas d'attaques	• Équilibre moyen de la consommation en énergie dans le cluster • Risque d'avoir des capteurs non couverts par les *cNodes* lors de certaines phases
Sélection selon l'énergie résiduelle	• Bonne répartition de l'équilibre • Surveillance de tous les capteurs par au moins deux *cNodes*	• L'ajout de *vNodes* est contraignant ; couteux en énergie et lourd à implémenter • Très gourmand en énergie
Élection démocratique	• Bonne répartition de l'équilibre • Surveillance de tous les capteurs par au moins deux *cNodes* • Consommation d'énergie modérée (après la période initiale) • Adaptation possible pour prendre en compte d'autres paramètres	• La période initiale consomme beaucoup d'énergie

TAB. 6.3 : Cas d'utilisation proposés pour les différentes méthodes de sélection

MÉTHODE	CAS D'UTILISATION
Aucun renouvellement	Fortement déconseillée de manière générale (la sécurité apportée ne vaut pas de mettre en place le dispositif, sauf en cas d'usage de capteurs dédiés du point de vue matériel).
Sélection aléatoire	Applications dont la sécurité n'est pas une priorité absolue, dont on veut garder des capteurs en vie très longtemps (mais pas nécessairement *tous* les capteurs).
Sélection selon l'énergie résiduelle	*A priori*, mieux vaut utiliser l'élection démocratique ; sauf en cas de courtes périodes d'activité du cluster.
Élection démocratique	Applications dont la sécurité est essentielle, et dont il faut maintenir en activité la totalité des capteurs. Déconseillée pour des applications imposant de courtes périodes d'activité au cluster.

3.2 Adapter le processus au réseau

3.2.1 ▶ Nombre des *cNodes*

Les résultats numériques de la section précédente, et notamment les différences de performances sur l'usage du processus d'élection démocratique pour désigner sept ou dix nœuds mettent en évidence l'importance du nombre de *cNodes* dans le cluster sur la préservation de l'énergie. Comme les *cNodes* consomment davantage, plus ils sont nombreux, plus les réserves se vident rapidement ; mais plus les chances de détecter l'attaque sont élevées. Il est donc intéressant, lors du déploiement du réseau, d'observer quel est le nombre idéal de *cNodes* à sélectionner pour assurer une couverture correcte sans gaspiller trop d'énergie. Dans notre cas, descendre à sept *cNodes* semble raisonnable, surtout avec des méthodes qui assurent une couverture minimale. En dessous de cette valeur, le risque de laisser des zones non couvertes devient important.

Le nombre de *cNodes* à utiliser est très dépendant de la topologie du réseau. Pour rappel, nous avons travaillé jusqu'à maintenant dans un cluster dont tous les nœuds sont des voisins directs du cluster head. Si les *cNodes* sont appelés à être déployés sur une autre architecture, sans cluster par exemple, les problèmes de couverture seront différents.

3.2.2 ▶ Contraintes sur la couverture du cluster

Un exemple déjà évoqué dans la sous-section 1.4 est celui d'un réseau en étoile, comme présenté en figure 6.10 : dans une telle configuration, il est indispensable d'adapter (ou de supprimer) la règle selon laquelle chaque capteur doit être surveillé par au moins deux *cNodes* (pour les méthodes d'élection démocratique et de sélection selon l'énergie résiduelle), sans quoi tous les nœuds ou presque devront être sélectionnés en tant que *cNodes*. Pour autant, cette règle n'est pas à rejeter systématiquement : elle est très utile dans la configuration avec cluster que nous avons étudié, et permet d'y limiter le nombre de *cNodes* à désigner.

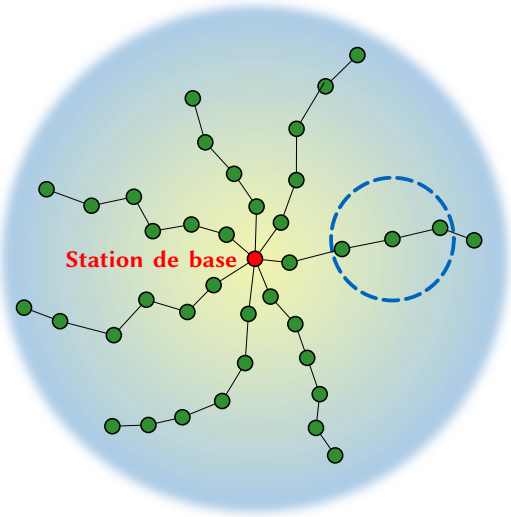

FIG. 6.10 : Schéma d'un réseau de capteurs en forme d'étoile. Le cercle en pointillés bleus représente la portée d'un nœud.

3.2.3 ▶ Interactions entre *cNodes* et nœuds compromis

La mise en place d'un système de détection efficace passe également par une implémentation rigoureuse de l'algorithme utilisé par les *cNodes* pour détecter les attaques. Aucune recommandation particulière n'est fournie, si ce n'est de définir en amont du déploiement un cahier des charges précis couvrant les cas d'utilisation et de fonctionnement normaux du réseau, de manière à pouvoir mettre en œuvre les règles de détection de la façon la plus efficace possible.

Conjugué à un renouvellement périodique selon une méthode de détection aléatoire ou d'élection démocratique, l'usage des *cNodes* permet une surveillance efficace du réseau. Il serait tout de même appréciable de pouvoir construire un modèle plus formel de cette surveillance. Puisque nous avons plusieurs acteurs avec des buts différents, les capteurs « sains » et les nœuds compromis, c'est vers des éléments de la théorie des jeux que nous allons nous tourner.

7

Modélisation à l'aide de jeux quantitatifs

Trois processus différents ont été présentés, dans les chapitres précédents, pour le renouvellement de *cNodes* afin d'assurer la détection des attaques dans un réseau de capteurs sans fil. En quelque sorte, ils ont été conçus de façon « incrémentale » : simple sélection aléatoire tout d'abord, qui pour améliorer l'équilibre de la consommation dans le réseau laisse place à la sélection selon l'énergie résiduelle ; à son tour celle-ci présente des défauts, que nous avons voulu corriger avec le processus d'élection démocratique. Lorsque nous en avons eu l'occasion, nous avons utilisé des outils de modélisation pour mettre en forme ces systèmes sous un aspect plus formel, et pour pouvoir en extraire certaines propriétés.

Dans ce chapitre, c'est la démarche inverse qui est mise en œuvre. Sans aller jusqu'à proposer un nouveau processus de sélection, nous avons souhaité modéliser le système en amont, afin de pouvoir, dans de futurs travaux, construire des protocoles efficaces à partir de l'architecture formelle obtenue. Pour ce faire, nous avons choisi de représenter le cluster sous la forme de jeux quantitatifs opposant plusieurs agents, qui cherchent à maximiser leur valeur de gain (correspondant à leur objectifs en termes d'envois de messages) et à conserver une énergie strictement positive.

Après une brève introduction à l'application de la théorie des jeux dans le contexte de la sécurité des réseaux de capteurs, nous définiront les jeux quantitatifs étudiés puis fourniront un exemple sur un premier tour de jeu ; enfin sera abordée leur résolution.

1 Théorie des jeux et réseaux de capteurs

1.1 Application au domaine de la sécurité

Nous avons évoqué la théorie des jeux au chapitre 3, en nommant certains outils de modélisation utilisés pour implémenter les méthodes de détection d'intrusion. Il semble utile à ce stade de fournir davantage d'éléments sur les liens qui rattachent cet outil à la sécurité des réseaux de capteurs sans fil.

La théorie des jeux est un ensemble d'outils permettant l'analyse de situations données, les « jeux », où les décisions prises par chaque entité (ou « joueur ») se basent sur l'anticipation des décisions des autres joueurs. Ces outils servent dans un premier temps à modéliser le jeu étudié, et consistent ensuite à résoudre des problèmes tels que la recherche d'équilibres ou de solutions optimales dans le jeu. La théorie des jeux est très fréquemment utilisée en économie. Elle intervient aussi régulièrement en politique, en biologie ou même en philosophie. Lui font appel, de manière plus ponctuelle, de nombreux autres domaines, dont l'informatique.

Appliquée aux réseaux de capteurs, une étude classe la théorie des jeux selon trois approches : les performances énergétiques, la sécurité, et les jeux de poursuite-évasion [85]. Les jeux de poursuite-évasion impliquent un ensemble de joueurs mobiles qui tentent de « capturer » un second ensemble de nœuds, d'en optimiser le suivi ; les agents du second groupe, pour leur part, cherchent à éviter de se faire détecter. Les deux autres catégories parlent d'elles-mêmes : les jeux appliqués aux performances énergétiques recherchent des solutions optimales permettant de minimiser la consommation, que ce soit en agissant sur la topologie du réseau ou bien sur le comportement des nœuds [29]. En sécurité, bien entendu, les jeux opposent les capteurs « normaux » aux agents attaquant de l'extérieur ou bien (dans le cas de nœuds corrompus) de l'intérieur du réseau.

Dans une seconde étude plus récente, les jeux de poursuite-évasion ne constituent qu'un élément d'une catégorie plus large regroupant les « applications » des capteurs (et comprenant aussi la collecte de données, par exemple). De même, les performances énergétiques ont été divisées en deux catégories, à savoir la gestion du réseau (ressources, énergie) et des communications (qualité de service, topologie, structure de routage) [114].

Mais revenons sur le domaine de la sécurité des réseaux, pour nous intéresser à quelques exemples d'applications. Le protocole LEACH a été sujet à de nombreuses propositions d'améliorations, comme nous l'avons déjà signalé au chapitre 3, et l'une d'entre elles a recours à la théorie des jeux [91]. Avec le protocole S-LEACH, la station de base utilise un système de détection d'intrusion centralisé tandis que les cluster heads utilisent des versions locales allégées, pour détecter les nœuds responsables de pertes de paquets dans le réseau. Les interactions entre les capteurs et l'IDS sont modélisées sous la forme d'un jeu bayésien (c'est-à-dire avec information partielle : l'IDS ne sait pas, en débutant la partie, si un nœud donné est compromis ou non). Chaque nœud se voit affecter un score de réputation. Les nœuds corrompus peuvent coopérer (pour éviter d'être détectés) ou bien supprimer des paquets. Les auteurs démontrent que le jeu possède deux équilibres de Nash, qui fournissent un moyen de détecter les nœuds compromis ou bien de les forcer à coopérer, rendant de fait l'attaque inefficace. Les jeux bayésiens ont aussi été utilisés dans des travaux de thèse [53], pour démon-

trer l'efficacité d'un autre système de détection d'intrusion ; mais dans ce deuxième exemple, ils interviennent davantage au niveau de la validation *a posteriori* du modèle plutôt qu'au moment de sa conception.

Les jeux répétés sont des modèles basés sur des séquences de stratégies qui dépendent de l'historique des actions déjà réalisées, où les décisions des joueurs vont se retrouver influencées par les actions passées de leurs adversaires et par les résultats intermédiaires qui en découlent. Ils peuvent facilement être rattachés à des mécanismes produisant des scores de réputation. Ces jeux sont donc employés pour créer des systèmes de détection d'intrusion génériques basés sur des mécanismes de confiance [3], ou bien sur des solutions plus particulières qui se concentrent sur l'usage d'accusés de réception dans le but de détecter un nœud compromis situé sur la route de retransmission des données [104].

1.2 Jeux quantitatifs, tours de jeux

Notre approche se distingue des travaux existants, dans le sens où l'arrêt potentiel (dû à l'épuisement de la batterie) est inclus dans les contraintes du jeu, en parallèle aux valeurs de gain. Pour autant que nous sachions, les travaux portant sur des configurations similaires considèrent toutes les dimensions comme étant soit liées à l'énergie, soit au gain, et traitent uniquement des conjonctions d'atomes [26, 131]. Dans de tels cas, les algorithmes sont établis sur une même structure : l'objectif du jeu est décomposé selon les dimensions établies, et l'on cherche pour chacune de ces dimensions une stratégie gagnante à mémoire finie. Ces stratégies sont ensuite combinées pour obtenir une stratégie globale pour le problème, qui elle peut éventuellement nécessiter une mémoire infinie.

D'autres analyses [130] traitent d'objectifs plus complexes basés sur la combinaison d'objectifs portant sur le gain moyen, en utilisant les opérateurs *somme*, *max* et *min*. Dépassant le cadre de nos travaux, on trouve aussi l'étude de jeux comportant à la fois des conditions sur le gain et un objectif de parité (où pour remporter la victoire, un joueur a besoin d'accéder au gain maximum qui sera récolté à l'infini sur le chemin, infini lui aussi, qui représente le déroulement de la partie) [25]. Dans ce cas l'objectif de parité assure que le système se comporte de façon cohérente, tandis que le gain représente un but quantitatif plus concret. Les stratégies gagnantes pour ces jeux peuvent nécessiter une mémoire infinie, mais il est possible d'en obtenir une approximation exécutable avec une mémoire finie. Ce type de jeux ne convient néanmoins pas parfaitement pour modéliser les réseaux de capteurs, car ils laissent de côté l'énergie, qui est un paramètre essentiel de notre contexte de travail.

Nous cherchons donc à combiner plusieurs objectifs, qui portent à la fois sur une valeur de gain et sur l'énergie restante dans la batterie des capteurs. Traditionnellement, un jeu se déroule tour par tour : chaque joueur joue sur son tour, puis laisse jouer ses concurrents (ou ses alliés selon le cas) ; il nous faut également composer avec cette contrainte, par exemple en définissant des intervalles de temps au cours desquels chaque joueur possède une action à réaliser.

2 Jeux quantitatifs infinis sur graphes finis

2.1 Modèle

Nous considérons dans la suite un réseau de capteurs sans fil, qui ne comporte pas nécessairement de clusters, déployé et en cours de fonctionnement. L'un (ou plusieurs) des capteurs a été compromis par un attaquant, et tente d'envoyer un maximum de messages, tout en cessant de relayer les messages de ses voisins. Les nœuds non compromis tentent de collaborer entre eux, et ils peuvent être vus comme membres d'une *coalition* dont le but est d'assurer le bon fonctionnement du réseau global. Le nœud compromis poursuit un objectif différent : il tente d'envoyer le plus de messages possibles. Tous les capteurs doivent faire face aux même contraintes énergétiques. L'envoi de données est plus couteux pour un capteur, en termes d'énergie, que de rester inactif : envoyer trop de données peut naturellement conduire à l'épuisement des réserves des nœuds.

2.2 Graphe du jeu

L'*arène* du jeu est un graphe $\mathcal{G} = (\mathcal{S}, \mathcal{A})$ où $\mathcal{S} = \mathcal{S}_\mathfrak{c} \uplus \mathcal{S}_\mathfrak{s}$ est l'ensemble des états, partitionné entre l'ensemble $\mathcal{S}_\mathfrak{s}$ des états des capteurs non compromis et les états $\mathcal{S}_\mathfrak{c}$ des capteurs compromis. L'ensemble $\mathcal{A}$ des arêtes est un sous-ensemble de $\mathcal{S} \times \mathcal{S}$ tel que pour tout $q \in \mathcal{S}, \exists q' \in \mathcal{S}, (q, q') \in \mathcal{A}$, autrement dit : il n'existe pas d'état final. Le graphe est *pondéré* sur k dimensions, ce qui signifie que l'on fait appel à une fonction $w : \mathcal{A} \to \mathbb{Z}^k$ qui assigne des poids pour chaque dimension considérée, à chaque arête du graphe.

2.3 La sémantique

Une *configuration* du jeu est un tuple $(q, p) \in \mathcal{S} \times \mathbb{Z}^k$: q est l'état actuel tandis que p est le gain accumulé. Depuis une configuration donnée (q, p), si l'on emprunte une arête $e = (q, q')$, alors la configuration suivante est $(q', p + w(e))$.

Une *exécution* est une séquence infinie de telles configurations. Un préfixe fini d'une exécution est appelé un *historique*. La partition de $\mathcal{S}$ détermine le joueur qui, depuis un état, choisira l'arête suivante.

Une *stratégie*, pour un joueur donné, est une fonction qui détermine le prochain état qui sera atteint (autrement dit, l'arête qui sera empruntée) : ainsi $\sigma_\mathfrak{c} : \mathcal{S}^*\mathcal{S}_\mathfrak{c} \to \mathcal{S}$ et $\sigma_\mathfrak{s} : \mathcal{S}^*\mathcal{S}_\mathfrak{s} \to \mathcal{S}$.

Si l'on considère une paire de ces stratégies ainsi qu'une configuration initiale donnée, le jeu produit une unique exécution appelée *résultat* des stratégies, et notée : $\mathit{issue}(\sigma_\mathfrak{c}, \sigma_\mathfrak{s})$.

Remarque 1. Toutes les notions définies ci-dessus s'étendent naturellement au cas des jeux impliquant plus de deux joueurs.

2.4 Conditions de victoire

Nous considérons ici des jeux à somme nulle : le but des capteurs « sains » n'est pas d'atteindre un gain fixé, mais de faire en sorte que les capteurs compromis perdent la partie. D'une manière générale, une *condition de victoire* est un sous-ensemble des exécutions du jeu. Pour des raisons pratiques, les conditions de victoire étudiées dans la littérature sont celles qui peuvent être exprimées de manière finie, aussi appelées conditions ω-régulières [46], et celles qui sont basées sur les valeurs de compteurs.

Dans notre cas, les dimensions relatives aux poids correspondent soit à un niveau d'énergie, soit à un gain. Elles peuvent donc être dissociées en deux ensembles : k_e dimensions relatives à l'énergie, et k_v relatives aux gains. Une configuration du jeu devient alors $(q, \mathit{val}, \mathit{énerg})$. Les niveaux d'énergie doivent rester supérieurs à zéro, quelles que soient leurs valeurs par ailleurs. Les gains peuvent être négatifs, bien que le but des joueurs soit de le maximiser. Comme l'on considère des exécutions infinies, il est plus pertinent de considérer la moyenne du gain sur la durée des exécutions, c'est-à-dire de prendre le *gain moyen* pour objectif.

L'ensemble des exécutions victorieuses pour un joueur est constitué des exécutions qui vérifient une *formule de gain* donnée, elle-même définie par la grammaire suivante :

$$\varphi ::= \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \neg \mathit{at} \qquad\qquad \mathit{at} ::= p_e \bowtie c \mid p_v \bowtie c$$

où :

- $c \in \mathbb{N}$;
- $\bowtie \in \{\geq, >\}$ est un opérateur de comparaison ;
- p_e est une composante relative à l'énergie ;
- p_v est une composante relative au gain.

La sémantique employée est la suivante :

- une exécution ρ satisfait un atome $p_e \bowtie c$ si pour chaque $i \in \mathbb{N}$, $\mathit{énerg}(\rho_i)_e \bowtie c$.
- une exécution ρ satisfait un atome $p_v \geq c$ si $\limsup_{n\to\infty} \frac{\mathit{val}(\rho_{\leq n})_v}{n} \geq c$.

Autrement dit, pour une composante donnée, on considère la somme de tous les poids divisée par le nombre d'étapes menées. Puisque les exécutions sont infinies, on considère la limite de la moyenne pour des préfixes de taille croissant vers l'infini. Comme la limite en elle-même n'existe pas toujours (si la valeur oscille, par exemple), nous choisissons la limite supérieure. Cela s'accorde avec l'idée que le réseau considère le pire scénario possible : le joueur gagne si, au pire, son gain moyen repasse périodiquement au-dessus de c lorsque la longueur du préfixe de l'exécution tend vers l'infini (bien que l'utilisation de la limite inférieure soit également possible, elle conduirait à la conception d'un modèle différent (d'une vision différente du problème), qui serait par ailleurs plus compliqué à résoudre).

La satisfaction d'une combinaison booléenne d'atomes est définie de la manière classique. La négation ne peut être appliquée que sur des atomes. On appelle *littéral* un atome ou sa négation. Lors de la résolution des jeux, on considèrera le *fragment positif* de cette logique, c'est-à-dire des conjonctions de littéraux. Une exécution devient ainsi gagnante si la formule est satisfaite, ce que l'on note $\rho \vDash \varphi$.

Les gains des joueurs débuteront avec la valeur 0, tandis que les niveaux en énergie seront crédités dès le début du jeu d'une valeur strictement positive, appelée *crédit initial*.

2.5 Problèmes de décision

Les problèmes de décisions qui sont étudiés dans ce contexte sont les suivants :

- **Problème de victoire** : étant données une configuration initiale ainsi qu'une formule de gain φ, existe-t-il une stratégie $\sigma_{\mathfrak{c}}$ pour le capteur compromis telle que pour toute stratégie $\sigma_{\mathfrak{s}}$ des capteurs « sains », $\mathit{issue}(\sigma_{\mathfrak{c}}, \sigma_{\mathfrak{s}}) \vDash \varphi$?
- **Problème du crédit initial** : étant donnés un état initial q ainsi qu'une formule de gain, existe-t-il une valeur $\chi \in \mathbb{N}$ telle que le problème de victoire basé sur la configuration $(q, 0^{k_p}, \chi^{k_e})$ ait une réponse positive ?

3 Exemple d'application

Pour illustrer concrètement le modèle présenté en section 2, nous décrivons le graphe $\mathcal{G}_{ex} = (\mathcal{S}, \mathcal{A})$ représentant le jeu pour un exemple de réseau.

3.1 Machine à états

Nous définissons quatre états distincts pour les capteurs :

- *au repos (repos)* : le nœud n'exécute aucune action ;
- *à l'écoute (écoute)* : le nœud est à l'écoute sur le canal de transmission, et reçoit éventuellement des données ;
- *traitement du message (trait_msg)* : le nœud place en mémoire tampon puis traite les paquets reçus ;
- *envoi de messages (envoi)* : le nœud envoie des données à l'un de ses voisins.

3.1.1 ▶ Nœud légitime

La machine à états pour un nœud légitime (non compromis) est présentée sur la figure 7.1. Pour les nœuds non compromis par un attaquant, l'énergie décroit lorsque le composant radio est utilisé, c'est-à-dire lors de l'écoute ainsi que lors de l'envoi de données. Lorsqu'un paquet est reçu, il est d'abord placé en mémoire tampon, avant de subir éventuellement des opérations de traitement (agrégation, compression, traitement sémantique) puis d'être transmis vers la station de base. Un nœud accumule du gain lorsqu'il envoie ses propres paquets. Il n'en accumule pas, en revanche, lorsqu'il ne fait que retransmettre des paquets (opération de routage) envoyés par un autre nœud.

Nous supposons ici qu'un capteur est capable de recharger sa batterie lorsqu'il reste au repos (c'est-à-dire lorsqu'il boucle sur l'état repos). Ce pourrait être le cas, par

exemple, de capteurs équipés de panneaux solaires. En pratique, la plupart des capteurs sont incapables de recharger leurs batteries, mais cette supposition nous permet ici de considérer des exécutions de durée infinie sur le jeu, ce qui serait impossible si l'énergie était strictement décroissante.

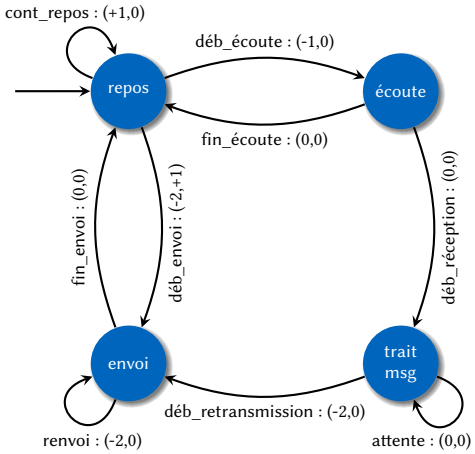

FIG. 7.1 : Machine à états pour les nœuds légitimes

3.1.2 ▶ Nœud compromis

Pour un nœud compromis, la machine à états est celle présentée en figure 7.2. Contrairement aux capteurs légitimes, un capteur compromis ne cherche pas nécessairement à retransmettre les paquets aux nœuds auxquels ils sont destinés. Aussi peuvent-ils revenir sur l'état repos sans retransmettre les paquets enregistrés en mémoire tampon.

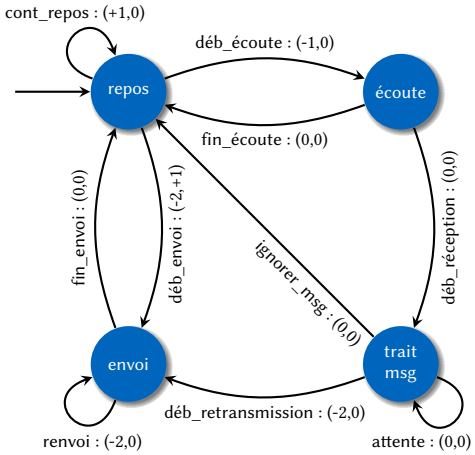

FIG. 7.2 : Machine à états d'un nœud compromis

3.2 L'arène

Nous considérons pour cet exercice un réseau contenant trois capteurs sans fil. L'un d'entre eux exactement a été compromis par un attaquant. Nous nommerons ces capteurs $\mathfrak{s}1$, $\mathfrak{s}2$ et $\mathfrak{c}3$, où $\mathfrak{c}3$ est le nœud compromis. Pour représenter le comportement asynchrone des nœuds sur le graphe, nous choisissons de découper le temps en intervalles. Nous considérons alors que pour chaque intervalle, chaque capteur a le temps d'exécuter exactement une action [1]. Sur chaque intervalle, les participants jouent de manière séquentielle (les uns après les autres) en se déplaçant sur le graphe. Nous choisissons un ordre arbitraire pour cette séquence : $\mathfrak{s}1$ joue en premier, puis vient le tour de $\mathfrak{s}2$, et enfin $\mathfrak{c}3$ passe en dernier. Comme les nœuds sont supposés jouer de façon *simultanée*, cet ordre utilisé pour la modélisation n'a aucune importance.

Le graphe qui représente le jeu possède un ensemble d'états symbolisant les différents états du réseau. Étant donné que chacun des trois joueurs dispose de quatre états (repos, envoi, écoute, trait_msg), le réseau dans son ensemble peut en théorie adopter jusqu'à $4^3 = 64$ états différents. En pratique, certains de ces états ne seront jamais atteints : par exemple, un état où tous les joueurs sont simultanément en train de traiter les paquets reçus n'aurait pas de sens, puisqu'il faudrait qu'au moins l'un d'entre eux ait émis ces paquets au tour précédent, et ce joueur devrait désormais se retrouver au repos. La forme donnée aux états du graphe de jeu indique quelle « équipe » s'apprête à jouer : les cercles sont utilisés pour les coups des capteurs légitimes, les rectangles servent pour les actions du nœud compromis [2].

Chaque arête du graphe représente l'action réalisée par le joueur qui a la main. Elles correspondent aux arêtes empruntées par les joueurs sur leurs propres machines à états, auxquelles seraient ajoutées des valeurs nulles en gain et en énergie pour les deux autres joueurs qui attendent leur tour. La figure 7.3 présente le premier tour du jeu pour chacun des trois joueurs (autrement dit, le premier intervalle de temps). Les états trait_msg ne peuvent pas être atteints avant le second tour de jeu, puisqu'ils requièrent que l'un des joueurs au moins soit passé auparavant par l'état écoute. Comme précisé plus haut, le graphe complet n'a pas d'état final.

3.3 Conditions de victoire

Plusieurs conditions de victoire peuvent être considérées. Le choix de la condition à retenir dépend du but que cherche à atteindre le nœud compromis.

3.3.1 ▶ Comportement cupide

Un capteur agissant de manière *cupide* cherche à envoyer le plus de messages possible, tout en restant en vie.

La formule de gain peut alors s'exprimer comme suit : $e_3 > 0 \wedge p_3 \geq \frac{1}{6}$, ce qui signifie dans cet exemple que pour gagner la partie le nœud compromis doit envoyer un message toutes les 6 étapes. Comme une action du nœud compromis n'intervient que toutes les 3 étapes seulement (les deux autres correspondant aux coups des

[1] À noter qu'une « action » correspond ici à un état, et peut entre autres consister à « rester au repos ».

[2] Cette convention n'est en fait pas toujours respectée sur la figure 7.3, car pour simplifier le graphe nous avons fusionné les états identiques (d'où les états possédant des arêtes bouclant sur eux-mêmes), et dans certains cas le joueur $\mathfrak{c}3$ est ainsi amené à agir depuis un état circulaire. En revanche, le code de couleur indiqué sur la figure pour les arêtes est respecté.

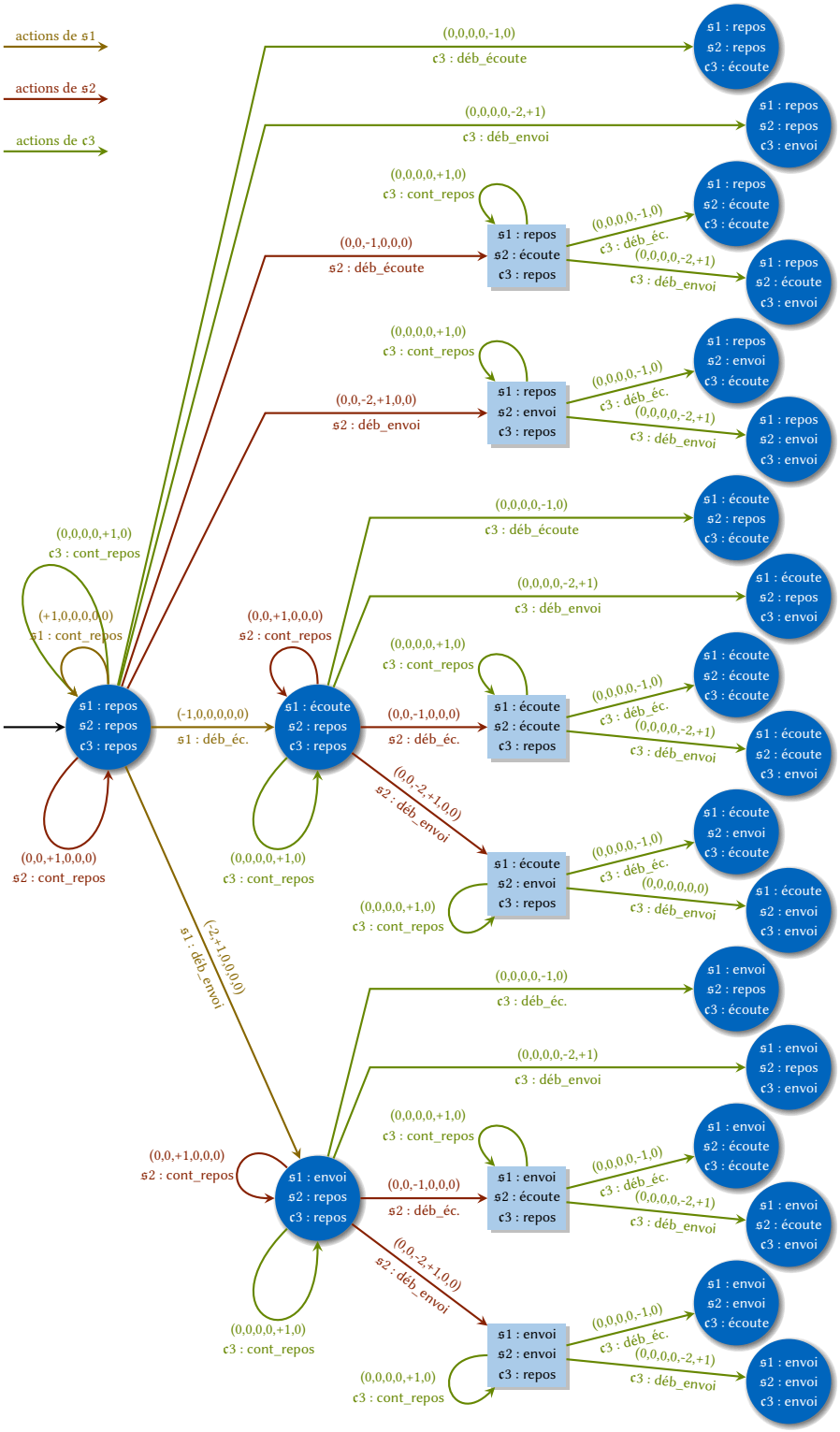

FIG. 7.3 : Sous-ensemble du graphe associé au jeu, représentant le premier tour

autres joueurs), le nœud compromis doit en fait emprunter une fois sur deux l'arête déb_envoi.

Dans ce cas il lui est en fait impossible de gagner, puisque l'envoi d'un message consomme ici 2 unités d'énergie, et la recharge de la batterie nécessite trop de « temps » pour permettre un envoi toutes les deux actions. L'objectif pourrait bien sûr être adapté à une autre définition d'un comportement « cupide » : nous pourrions considérer, par exemple, que l'envoi d'un message toutes les 15 étapes peut remplir l'objectif d'un nœud cupide.

3.3.2 ▶ Comportement destructeur

Un capteur se comportant de façon *destructrice* cherche à rendre le réseau inutilisable, par exemple en agissant de telle façon que tous les nœuds épuisent leur batterie : son but est alors de demeurer le dernier nœud en état de marche dans le réseau. La formule de gain s'écrit dans ce cas $e_3 > 0 \wedge e_1 \leq 0 \wedge e_2 \leq 0$.

3.3.3 ▶ Comportement perturbateur

Un capteur adoptant un comportement *perturbateur* cherche à empêcher ses voisins de transmettre leurs messages, tout en cherchant encore une fois à rester en état de marche. Son objectif n'est donc plus de vider la batterie des nœuds, mais seulement de rendre ceux-ci inutiles, de les empêcher d'accumuler du gain. La formule de gain devient alors $e_3 > 0 \wedge (e_1 \leq 0 \vee p_1 \leq 0) \wedge (e_2 \leq 0 \vee p_2 \leq 0)$.

4 Résolution des jeux

Cette section s'intéresse au problème posé par la résolution de ces jeux. Nous montrerons tout d'abord que la résolution est un problème indécidable lorsque la condition de victoire est une formule de gain arbitraire. Nous nous concentrerons ensuite sur le segment positif de ces formules, et donnerons des conditions suffisantes pour pouvoir résoudre le problème.

Afin de clarifier la présentation, nous considérons dans cette section deux joueurs, simplement désignés par les termes joueur 0 et joueur 1. Le joueur 0 est celui qui cherche à atteindre l'objectif décrit par la formule de gain, et représente donc le nœud compromis. Conformément à la convention adoptée dans la section précédente, ses états seront représentés sous forme de rectangles. Le joueur 1 quant à lui représente la coalition (l'ensemble) des autres nœuds, et ses états seront de forme circulaire.

4.1 Indécidabilité du cas général

Nous allons démontrer ici que le problème de victoire pour les jeux étudiés ne présente pas de solution algorithmique si la condition de victoire ne peut pas être spécifiée par une formule de gain. Le théorème établi est le suivant :

Théorème 1. Le problème de victoire, ainsi que le problème de crédit initial, sont indécidables pour des objectifs définis par une formule de gain comportant quatre composantes relatives à l'énergie et une composante relative au gain.

La preuve de ce théorème consiste à encoder, à l'aide d'un jeu du type étudié, le problème de terminaison à l'aide d'une machine à deux compteurs : le problème équivalent ainsi obtenu est connu comme étant indécidable [89]. Chaque compteur est représenté par deux composantes d'énergie (une pour chacun des deux joueurs), qui contiennent chacune une copie de la valeur du compteur en question. Pour l'instruction *0-test* (présentée ci-dessous), le joueur 0 peut prétendre que son compteur est nul, ou bien qu'il ne l'est pas ; le second joueur peut vérifier si cette affirmation est valide ou non. Si la machine atteint son état d'*arrêt*, une composante de gain est incrémentée, et tous les autres composantes sont réinitialisées à 0. De cette manière, si la machine s'arrête, le joueur 0 a une stratégie gagnante lui permettant d'obtenir un gain strictement positif. Autrement, le gain demeure nul.

On définit plus précisément une machine à deux compteurs comme une liste d'instructions étiquetées, qui peuvent être :

- *incrément :* incrémenter le compteur c (respectivement d) et aller à l'étiquette ℓ ;
- *0-test :* si $c = 0$ (respectivement $d = 0$) alors aller à l'étiquette ℓ_1, sinon décrémenter c (respectivement d) et aller à l'étiquette ℓ_2 ;
- *arrêt :* arrêt de la machine, associé à l'état halte du jeu.

Au départ, les deux compteurs sont initialisés à 0.

À présent, étant donné une machine $\mathcal{M}$, nous construisons une arène $\mathcal{G}_{\mathcal{M}}$ basée sur des modules « élémentaires » pour chaque type d'instructions (celui du compteur c sera symétrique à celui du compteur d). L'arène comporte donc cinq composantes c, c', d, d', p, où seule p concerne le gain, les autres étant attachées à des valeurs énergétiques. Les composantes c et c' (respectivement d et d') sont supposées toujours conserver la même valeur, excepté pour l'instruction *0-test*. Au lancement du jeu toutes les composantes ont pour valeur [3] 0.

Le but du joueur 0 est d'obtenir un gain strictement positif, tout en simulant fidèlement la machine. Il faut entendre par cette expression que les valeurs des composantes c et d doivent demeurer positives. Les composantes c' et d' permettent de s'assurer que le joueur 1 ne pourra prétendre à tort que le joueur 0 a triché. Si le cas devait se produire, les valeurs de ces composantes seraient strictement négatives. À présent la condition de victoire s'exprime comme la satisfaction de la formule de gain suivante :

$$(c \geq 0 \wedge d \geq 0 \wedge p > 0) \vee (c' < 0 \wedge c \geq 0) \vee (d' < 0 \wedge d \geq 0)$$

Chaque instruction étiquetée a un état dans l'arène appartenant au joueur 0, avec la même étiquette (d'autres états seront rajoutés par la suite). Le cas de l'*incrément* est trivial : si l'instruction ℓ est « incrémenter c puis aller à l'étiquette ℓ' », alors il y a une arête partant de ℓ et joignant ℓ' avec les poids $(1, 1, 0, 0, 0)$. Il n'y a pas d'autre arête partant de ℓ, le joueur n'a donc pas d'autre choix.

L'instruction *0-test* est plus délicate, car le joueur 0 est susceptible de tricher en empruntant la mauvaise branche de l'instruction conditionnelle. De manière formelle, on suppose que ℓ est « si $c = 0$ alors aller à l'étiquette ℓ_1, sinon décrémenter c et aller à l'étiquette ℓ_2 ». Il y a alors deux arêtes sortant de l'état ℓ : l'une correspondant à l'annonce $c = 0$, l'autre à l'annonce $c > 0$, comme indiqué sur la figure 7.4. L'arête

[3] Pour obtenir une simulation plus réaliste, il est possible d'assigner aux composantes la valeur initiale 1, et de considérer 0 comme valeur de défaite. Nous choisissons ici d'autoriser un niveau d'énergie nul et considérons le jeu comme « perdu » pour des valeurs strictement négative, et ce afin de rendre plus clair le lien entre les valeurs des compteurs et les niveaux d'énergie.

portant l'affirmation $c = 0$ atteint un état possédé par le joueur 1, qui peut soit accepter l'affirmation (et aller dans ce cas à ℓ_1), soit la rejeter. Le rejet de cette affirmation implique de décrémenter c' (mais non c) et de se rendre sur un état arrêt : cet état possède une unique arête sortante, qui boucle sur lui-même avec les poids $(0, 0, 0, 0, 0)$, signifiant que la simulation s'est arrêtée. Si l'affirmation est rejetée à tort, on obtient $c' < 0$ tandis que $c = 0$, puisque les valeurs de c et c' étaient égales. Si l'affirmation a été rejetée à raison, alors le gain est nul.

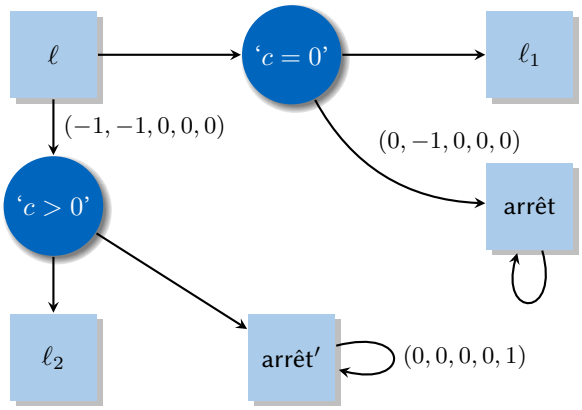

FIG. 7.4 : Module d'encodage de l'instruction *0-test* sous forme de jeu : « si $c = 0$ alors aller à l'étiquette ℓ_1, sinon décrémenter c et aller à l'étiquette ℓ_2 ». Les états sous forme de carrés appartiennent au joueur 0, les cercles représentent les états du joueur 1. Il est convenu qu'en l'absence d'indication de poids, les arêtes comportent les poids $(0, 0, 0, 0, 0)$.

De même, l'affirmation $c > 0$ est représentée par une arête qui décrémente à la fois c et c' (donc de poids $(-1, -1, 0, 0, 0)$), et atteint l'état du joueur 1. Ce dernier peut soit poursuivre la simulation, soit se rendre sur un état arrêt′. Cet état possède une unique arête qui boucle sur lui-même avec les poids $(0, 0, 0, 0, 1)$, ce qui signifie que la simulation est arrêtée mais que le gain vaut 1. Si le joueur 0 a triché, alors $c < 0$ et la simulation n'a pas besoin d'être menée plus avant. S'il s'est comporté de façon loyale, alors $c > 0$ et $c' > 0$, et $p > 1$.

Enfin, lorsqu'il atteint l'état halte, le joueur 0 décrémente à la fois c et c' autant de fois qu'il le souhaite, puis se rend sur un état de vérification (vérif). L'état vérif appartient au joueur 1, et agit de la même façon que pour le *0-test* dans le cas où le joueur 0 affirme que $c = 0$. On procède de façon identique pour d. Enfin, une arête retourne depuis l'état vérif jusqu'à l'état initial et rapporte une récompense à la composante de gain p : son poids est $(0, 0, 0, 0, 1)$. Ce module est représenté sur la figure 7.5.

Supposons à présent que $\mathcal{M}$ s'arrête. Considérons la stratégie du joueur 0 qui consiste à jouer honnêtement, c'est-à-dire à annoncer les valeurs correctes du compteur (et à décrémenter les composantes d'énergie, de sorte qu'elles atteignent exactement 0). Alors si le joueur 1 ne lance aucune accusation de tricherie, les compteurs c et d restent tous deux positifs, comme pour $\mathcal{M}$. Supposons de plus que $\mathcal{M}$ a terminé son exécution en k étape, et dénotons par c_k et d_k les valeurs respectives de c et d lorsque l'on atteint les instructions d'arrêt. Dans ce cas l'exécution de la simulation sur la machine correspond au plus à $2k$ étapes du jeu (à cause des affirmations qui

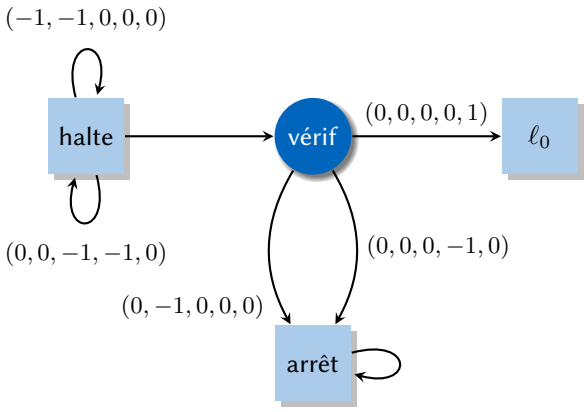

FIG. 7.5 : Module d'encodage de l'instruction d'*arrêt* sous forme de jeu. Les états sous forme de carrés appartiennent au joueur 0, les cercles représentent les états du joueur 1.

doivent être acceptées), et le jeu nécessite $c_k + d_k + 2$ étapes depuis l'état halte pour atteindre à nouveau l'état initial : il s'agit du nombre d'étapes requises pour décrémenter chaque compteur et pour traverser le module de la figure 7.5. En conséquence la valeur minimale du gain moyen est de $\frac{1}{2k+c_k+d_k+2} > 0$.

Si le joueur 1 a rejeté l'une des affirmations du joueur 0 :

- si $c = c' = 0$, mais que le joueur 1 a depuis décrémenté c' pour contester l'annonce, alors le jeu « s'arrête », mais la condition $c' < 0 \wedge c \geq 0$ est satisfaite ;
- si $c = c' \geq 0$ après le décrément mais que le jeu a été « arrêté », le gain vaut 1 tandis que les deux composantes c et d sont supérieures à 0. Dans ce cas la condition $c \geq 0 \wedge d \geq 0 \wedge p > 0$ est satisfaite.

Au final, on retient que la stratégie de simulation « loyale » est gagnante.

Supposons maintenant que $\mathcal{M}$ ne s'arrête pas : la simulation « loyale » retourne une valeur de gain de 0 si le joueur 1 ne porte aucune accusation à tort (l'état d'arrêt halte n'est jamais atteint, et si le joueur 1 ne réfute aucune annonce, l'état arrêt′ n'est pas non plus atteint.) Dans le cas où le joueur 0 « ment » lors de ses annonces, il suffit au joueur 1 de détecter cette erreur pour que l'on obtienne soit $c < 0$, soit une valeur de gain définitivement nulle.

Cette construction peut être adaptée au problème du crédit initial : il suffit seulement de rajouter, en amont de l'état initial, un module similaire au module halte dans le sens où il doit permettre à tous les niveaux d'énergie d'atteindre 0. Une fois l'état « initial » (à l'origine) atteint, le crédit initial devient caduque, et le joueur 0 gagne si et seulement si $\mathcal{M}$ s'arrête.

4.2 Le fragment positif

Revenons à présent sur le problème du crédit initial, dont le jeu est plus « simple » à remporter pour le nœud compromis que le jeu du problème de victoire correspondant. De plus il est courant, lorsque l'on combine des stratégies, qu'un crédit initial

nécessaire à la victoire soit augmenté ; en d'autres termes, les stratégies concernant le problème de victoire ne sont souvent pas assez robustes. Du point de vue du modèle employé, résoudre le problème du crédit initial permet d'obtenir plus d'informations : si une stratégie et un crédit sont découverts, ils fournissent une valeur concrète à laquelle l'énergie doit être initialisée.

Nous démontrons dans cette section que les stratégies gagnantes pour les objectifs définis par des littéraux peuvent être combinés en une stratégie gagnante pour l'objectif global.

Rappelons avant tout que le cas d'un littéral a été étudié en détail, pour des jeux relatifs soit à de l'énergie, soit à des gains moyen. Ces jeux ont une solution simple, dans le sens où :

- l'un des joueurs a une stratégie gagnante ;
- si une stratégie gagnante existe, alors il en existe une qui repose sur une mémoire finie ;
- déterminer quel sera le joueur vainqueur peut être réalisé en NP ∩ coNP [144].

De plus, il a été prouvé [131] qu'une mémoire finie est suffisante pour gagner sur une conjonction de contraintes sur l'énergie. Le cas des conjonctions de contraintes sur le gain a lui aussi fait l'objet d'études dans [131], où les stratégies sans mémoire pour chaque condition sur le gain sont combinées en stratégies à mémoire infinie pour l'objectif global. Par exemple, lorsque le gain moyen est défini par la limite supérieur du gain, chaque stratégie peut être jouée au cours de phases de durée croissante jusqu'à atteindre (ou à s'approcher aussi près que nécessaire de) la valeur désirée.

Par conséquent nous allons nous concentrer ici sur l'association d'objectifs portant à la fois sur l'énergie et sur le gain moyen. Il est bon de noter avant tout que si l'objectif du nœud compromis est une conjonction de littéraux, et que l'un deux ne peut être réalisé, il est évident que le jeu ne peut être remporté.

S'il existe des stratégies pour chacun des sous-objectifs, il peut être possible de combiner des stratégies gagnantes à mémoire finie pour chacun des littéraux en une unique stratégie gagnante, nécessitant potentiellement une mémoire infinie. Nous allons fournir des conditions suffisantes pour obtenir une telle stratégie globale ; la recherche algorithmique de la stratégie en question est un problème non résolu.

4.2.1 ▶ Attracteurs

Dans les jeux à deux joueurs, il est courant de faire appel à la notion d'*attracteurs* d'un ensemble $\mathcal{Q}$ pour désigner les états depuis lesquels un joueur peut forcer le jeu à se déplacer dans $\mathcal{Q}$.

Définition 1. Les *attracteurs en une-étape* d'un ensemble d'états $\mathcal{Q}$ pour les joueurs 0 et 1 sont définis ainsi :

$$\begin{aligned} 1Attr_0(\mathcal{Q}) &= \{q \in \mathcal{S}_0 \mid \exists (q,q') \in \mathcal{A} \text{ tel que } q' \in \mathcal{Q}\} \\ &\quad \cup \{q \in \mathcal{S}_1 \mid \forall (q,q') \in \mathcal{A}, q' \in \mathcal{Q}\} \end{aligned}$$

$$\begin{aligned} 1Attr_1(\mathcal{Q}) &= \{q \in \mathcal{S}_0 \mid \forall (q,q') \in \mathcal{A}, q' \in \mathcal{Q}\} \\ &\quad \cup \{q \in \mathcal{S}_1 \mid \exists (q,q') \in \mathcal{A} \text{ tel que } q' \in \mathcal{Q}\} \end{aligned}$$

Les *attracteurs* pour le joueur 0 (respectivement pour le joueur 1) d'un ensemble $\mathcal{Q}$ d'états sont les points fixes des attracteurs en une-étape, en partant de $\mathcal{Q}$: on a ainsi, pour $i \in \{0, 1\}$,

$$Attr_i(\mathcal{Q}) = \bigcup_{j \in \mathbb{N}} (1Attr_i)^j (\mathcal{Q})$$

L'attracteur pour le joueur i est donc l'ensemble des états depuis lesquels ce joueur est en mesure de s'assurer que le jeu atteindra $\mathcal{Q}$. On notera que pour tout état de $(1Attr_i)^j(\mathcal{Q})$, le joueur i a une stratégie sans mémoire lui permettant d'atteindre $\mathcal{Q}$ en au plus j étapes. Puisque le point fixe est atteint en $|\mathcal{S}|$ itérations au plus, $\mathcal{Q}$ peut être atteint avec un maximum de $|\mathcal{S}|$ étapes depuis n'importe quel état de $Attr_i(\mathcal{Q})$. On notera aussi que cette borne entraine la possibilité de calculer les attracteurs en un temps polynomial.

Lemme 2. Depuis n'importe quel état de $Attr_i(\mathcal{Q})$, le joueur i possède une stratégie sans mémoire qui l'assure d'atteindre $\mathcal{Q}$ en un maximum de $|\mathcal{S}|$ étapes.

Une propriété des attracteurs dans les jeux est qu'ils peuvent être retirés sans danger du jeu, tout en laissant la structure du graphe restant toujours associée à un jeu (c'est-à-dire sans état final) :

Lemme 3. Soit $\mathcal{G} = \langle \mathcal{S}_0, \mathcal{S}_1, \mathcal{A} \rangle$ le graphe d'un jeu (c'est-à-dire tel que tout état de $\mathcal{S}$ a une arête sortante appartenant à $\mathcal{A}$). Soit un joueur $j \in \{0, 1\}$. Soit $\mathcal{Q} \subseteq \mathcal{S}$. Considérons le graphe $\mathcal{G}' = (\mathcal{S}'_0, \mathcal{S}'_1, \mathcal{A} \cap (\mathcal{S}' \times \mathcal{S}'))$ avec $\mathcal{S}'_i = \mathcal{S}_i \setminus Attr_j(\mathcal{Q})$ pour $i \in \{0, 1\}$. Alors tout état de $\mathcal{S}'$ a une arête sortante allant dans $\mathcal{A} \cap (\mathcal{S}' \times \mathcal{S}')$, autrement dit $\mathcal{G}'$ est aussi un graphe de jeu.

Démonstration. Supposons par contradiction que ce ne soit pas le cas, c'est-à-dire qu'il existe $q \in \mathcal{S}'$ tel que q n'a aucune arête sortante. Comme q avait une arête sortante appartenant à $\mathcal{A}$ dans le graphe de jeu $\mathcal{G}$, cela signifie que tous les états successeurs de q appartiennent à $Attr_j(\mathcal{Q})$. Par définition d'un attracteur, on a alors $q \in Attr_j(\mathcal{Q})$, et on en déduit ainsi que $q \notin \mathcal{S}'$. □

4.2.2 ▶ Conjonction d'un objectif de gain et d'un objectif d'énergie

Dans ce premier exemple simple, nous étudions les objectifs, pour le joueur 0, de la forme $p_e \geq c_e \wedge p_v \geq c_v$, où p_e et p_v sont le niveau d'énergie et la valeur du gain moyen accumulés par le joueur 0, et avec $c_e, c_v \in \mathbb{N}$. De tels objectifs se traduisent par le besoin d'atteindre une certaine valeur de gain tandis que le niveau d'énergie demeure supérieur à une limite donnée. Un comportement cupide est une illustration simple d'un objectif de ce type : le nœud compromis cherche à rester « en vie » (donc à vérifier à tout instant $p_e \geq 1$) tout en émettant au moins un message toutes les 6 étapes ($p_v \geq \frac{1}{6}$, mais comme nous l'avons remarqué plus haut, il est possible d'obtenir un seuil à valeur entière en multipliant tous les poids de la composante par 6).

Tout d'abord, nous pouvons clairement établir que depuis un état à partir duquel au moins un des objectifs ne peut être rempli, le joueur 0 ne peut pas gagner. De plus, si le joueur 1 peut forcer le joueur 0 à atteindre l'un de ces états, alors l'objectif du joueur 0 ne sera pas rempli. De là nous pouvons utiliser la notion classique d'*attracteurs* dont nous avons rappelé la définition. Nous notons $\mathcal{P}_e$ les états où le joueur 0 perd pour $p_e \geq c_e$ (c'est-à-dire où le joueur 1 possède une stratégie pour l'empêcher de gagner), et $\mathcal{P}_v$ les états pour lesquels le joueur 0 perd pour $p_v \geq c_v$. Le joueur 1 peut évidemment empêcher le joueur 0 d'atteindre son objectif

$p_e \geq c_e \wedge p_v \geq c_v$ depuis tout état de $Attr_1(\mathcal{P}_e \cup \mathcal{P}_v)$. Par conséquent toute stratégie gagnante pour le joueur 0 doit rester dans l'ensemble $\mathcal{G} \setminus Attr_1(\mathcal{P}_e \cup \mathcal{P}_v)$. Par ailleurs, une stratégie gagnante qui reste dans $\mathcal{G} \setminus Attr_1(\mathcal{P}_e \cup \mathcal{P}_v)$ est aussi une stratégie gagnante dans $\mathcal{G}$ puisque le joueur 1 ne peut pas forcer le jeu à se déplacer dans $Attr_1(\mathcal{P}_e \cup \mathcal{P}_v)$ (par définition des attracteurs).

Ainsi il devient possible de retirer, de manière récursive, des états de $\mathcal{G}$ jusqu'à ce que le joueur 0 gagne pour chacun de ses objectifs depuis tous les états restants. Si, en revanche, le graphe de jeu devient vide à un moment donné, cela signifie que le joueur 0 ne peut pas gagner pour les deux objectifs depuis quelque état que ce soit du graphe d'origine.

Supposons à présent que le joueur 0 dispose des stratégies gagnantes λ_e, λ_v pour les objectifs respectifs $p_e \geq 1$ et $p_v \geq \frac{1}{15}$, et que ces stratégies sont gagnantes depuis tous les états. Ces stratégies sont supposées être sans mémoire, ce sont donc des fonctions de $\mathcal{S}_c$ dans $\mathcal{S}$.

Considérons le jeu à un joueur $\mathcal{G}_e$, obtenu quand le joueur 0 joue λ_e : chaque arête entrante dans un état $v \in \mathcal{S}_c$ va à la place à $\lambda_e(v)$, et les poids sont cumulés : $v_0 \xrightarrow{w_1} v \xrightarrow{w_2} \lambda_e(v)$ est remplacé par $v_0 \xrightarrow{w_1+w_2} \lambda_e(v)$. De façon similaire, soit $\mathcal{G}_v$ le jeu à un joueur obtenu en jouant λ_v.

Soit α la valeur la plus basse pouvant être obtenue pour la composante k_v sur un cycle simple sur $\mathcal{G}_e$, et soit de même β la valeur la plus basse pouvant être obtenue pour k_e lors d'un cycle sur $\mathcal{G}_v$. Plus concrètement, α représente la pire valeur de gain que le joueur peut obtenir lorsqu'il joue selon la stratégie qui l'assure de conserver le niveau d'énergie requis, tandis que β est la pire valeur en énergie qui puisse être atteinte en jouant la stratégie qui assure le gain requis. Il est à noter que si $\alpha \geq c_v$, alors λ_e est aussi une stratégie gagnante pour l'objectif $p_v \geq c_v$, et constitue donc une stratégie gagnante pour l'objectif global. De même si $\beta > 0$, alors λ_v vérifie l'objectif $p_e \geq c_e$ pourvu que le crédit initial soit adapté à la somme minimale des poids des arêtes empruntées au cours d'un cycle :

$$\text{crédit initial} = 1 + c + \min_{\substack{\rho \text{ préfixe de } \mathcal{C} \\ \mathcal{C} \text{ cycle dans } \mathcal{G}_v}} w_e(\rho)$$

Si la condition suffisante que nous venons de présenter n'est pas remplie, nous n'avons pas, à ce jour, de solution pour résoudre ces jeux dans un contexte général. Des stratégies gagnantes pour chaque composante pourraient en effet se révéler incompatibles entre elles. Prenons, pour illustrer, l'exemple du jeu à un seul joueur présenté en figure 7.6. Dans cet exemple, q_0 agit comme un état de recharge (en énergie) tandis que q_1 est un état actif qui produit un effet utile récompensé par une valeur de gain. On constate que recharger la batterie est bien plus lent que de consommer l'énergie, puisqu'il faut 42 « unités d'énergie [4] » par étape active.

Il est possible de maintenir un niveau d'énergie constamment positif (pour la première composante) en restant dans q_0 (où en y allant si l'on commence en q_1). Il est aussi possible d'atteindre un gain moyen de 1 en demeurant sur q_1 (ou en s'y rendant si l'on démarre sur q_0, car le poids de l'unique arête empruntée est négligeable sur une longue exécution). Cependant, il n'est pas possible de conserver un gain moyen de 1 tout en maintenant un niveau d'énergie positif, puisqu'une fois le crédit initial écoulé,

[4] Ce peut être, par exemple, une joule, ce qui signifierait que l'étape active consommerait avec une puissance de 42 W, tandis que la puissance de la recharge serait de 1 W. La valeur 42 a été choisie arbitrairement pour cet exemple.

42 étapes de recharge doivent impérativement être effectuées avant de réaliser toute action susceptible de rapporter un gain.

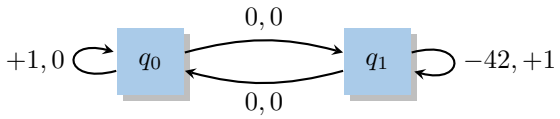

FIG. 7.6 : Un exemple de jeu simple nécessitant une mémoire infinie. La première composante est un niveau d'énergie, la seconde est une valeur de gain.

4.2.3 ▶ Avec mémoire limitée

Nous considérons que les réseaux de capteurs sans fil sont fortement limités en ressources disponibles, et ne peuvent donc implémenter que des stratégies à mémoire finie. Dans ce cas, limiter *a priori* la quantité de mémoire qui peut être utilisée par le joueur fournit une solution pour résoudre le problème du crédit initial pour des jeux ayant une condition de victoire sur le fragment positif.

Une stratégie à mémoire finie est une stratégie qui peut être implémentée par une machine de MEALY finie et déterministe : état donné l'état actuel de la machine et du jeu, la machine fournit l'arête à emprunter et de là l'état suivant de la machine. La taille de machine de MEALY est la taille de la mémoire disponible.

Par exemple, une stratégie à mémoire finie pour le jeu de la figure 7.6, qui boucle 42 fois sur q_0 puis se rend sur q_1, boucle une fois et enfin retourne sur q_0 pour recommencer le cycle, est représentée sur la machine de la figure 7.7. Cette machine possède 46 états, puisqu'elle nécessite de compter (et donc garder en mémoire) le nombre de fois où l'on a bouclé sur q_0 pour accumuler de l'énergie. Si l'on devait démarrer en q_1, la machine se rendrait tout d'abord en q_0 avant d'appliquer la stratégie que nous venons de décrire. La machine doit être complète au niveau des entrées possibles, aussi faut-il autoriser q_1 à survenir depuis n'importe quel état mémoire (dans notre cas elle retourne à l'état mémoire initial, tandis que le jeu retourne en q_0).

Remarque 2. Les stratégies pour une quantité finie donnée de mémoire peuvent ne pas être optimales. Considérons par exemple le jeu de la figure 7.6 avec pour objectif le maintien du niveau d'énergie au-dessus de 0 et d'obtenir un gain moyen supérieur ou égal à $\frac{1}{43}$.

Une stratégie à mémoire infinie pourrait remporter la victoire sur ce jeu. Elle serait découpée en phases qui se dérouleraient de la manière suivante : à la $k^{\text{ième}}$ phase, il faudrait boucler $42k$ fois sur q_0, puis se rendre en q_1 ; il faudrait alors boucler k fois avant de retourner sur q_0. Cela permettrait de faire en sorte que les poids des transitions entre q_0 et q_1 soient négligeables, et d'obtenir une limite, pour le gain moyen, de $\frac{1}{43}$, puisque 43 étapes seraient nécessaires pour incrémenter le compteur de gain d'une unité. De plus, à chaque phase, le niveau d'énergie retombe à sa valeur initiale, puis ne prend que des valeurs supérieures à celle-ci.

En revanche, si seulement m états de mémoire sont autorisés, la meilleure valeur de gain moyen que l'on peut espérer obtenir est de $\frac{m}{43m+2}$ (la stratégie correspondante consiste à répéter la phase m). Ainsi, avec une mémoire finie, non seulement le gain moyen optimal ne peut pas être atteint, mais on constate de plus que l'allocation de davantage de mémoire permet d'atteindre une meilleure valeur de gain moyen.

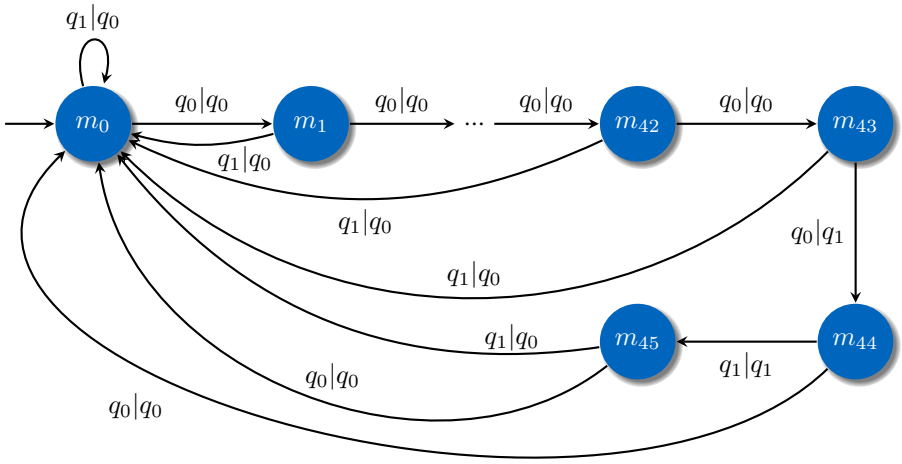

FIG. 7.7 : Une machine de MEALY représentant une stratégie à mémoire finie (46 états). L'extrémité sortante d'une arête représente l'état actuel du jeu, l'extrémité sortante est l'état suivant retenu.

Résoudre le jeu en supposant que le joueur 0 a une mémoire finie limitée à un nombre k d'unités fourni en entrée revient à deviner sa stratégie sous forme de machine de MEALY, ce qui constitue un objet de taille exponentielle si k est fourni sous forme binaire. La machine est alors synchronisée avec le jeu, et retourne un second jeu à un seul joueur, à faire jouer par le joueur 1. Dans ce second jeu, seules les stratégies sans mémoire doivent êtres prises en compte.

En effet, tout chemin infini qui ne satisferait pas $p_e \geq c_e \wedge p_v \geq c_v$ vérifierait, à un moment ou l'autre, soit $p_e < c_e$, soit $p_v < c_v$ (en considérant la limite pour le gain moyen de p_v). Dans les deux cas cela revient à trouver un chemin en « lasso » dans le graphe, ce qui peut être deviné.

En ce qui concerne la complexité, la procédure que nous venons de décrire est dans NEXPSPACE (qui est équivalent à EXPSPACE [97, Chap. 20]), même si la borne n'est pas atteinte.

4.3 Des jeux aux méthodes de détection

Nous avons modélisé le réseau de capteurs sans fil sous forme de jeux quantitatifs basés à la fois sur des valeurs de gain et d'énergie, dans lequel évoluent un nœud compromis ainsi qu'une équipe de capteurs sains. L'arène du jeu, les actions de chaque équipe, ainsi que la grammaire des conditions de victoire ont été spécifiées, avant que ne soit fourni un exemple détaillé d'un tel jeu.

Nous venons à présent de nous pencher sur la résolution des problèmes de victoire et de crédit initiale, qui n'est possible que pour des formules de gain définies sous forme de conjonctions de littéraux. L'indécidabilité du cas général montre bien que l'étude des interactions entre capteurs légitimes et nœuds compromis n'est pas triviale. La résolution sur le segment positif est toutefois un cas intéressant, qui permet de déterminer des conditions de victoire, ici pour le nœud compromis (c'est-à-dire une situation dans laquelle il sera à même de mener son attaque sans être détecté).

Les travaux menés à ce jour sur cette modélisation du réseau sous forme de jeux quantitatifs s'arrête ici. Nous envisageons de poursuivre leur étude afin de pouvoir, à terme, obtenir des jeux, des conditions de victoire et des résolutions suffisamment poussés pour nous permettre d'établir de nouvelles méthodes de détection ou de prévention des attaques par déni de service dans le réseau, en forçant un nœud compromis à coopérer avec les capteurs alentour sous peine de perdre la partie.

Avec ce chapitre se referment les aspects techniques présentés dans cet ouvrage ; il ne nous reste plus à présent qu'à tirer les conclusions des travaux présentés, et à introduire les perspectives envisagées pour la suite des recherches.

8

Conclusion

1 Rappel des enjeux

Aussi bien dans les domaines public, industriel, militaire que chez les particuliers, la volonté de collecter puis d'analyser des données sur l'environnement, sur des processus, sur des flux, n'a de cesse de croitre. Poussés par les progrès constants dans la miniaturisation des composants électroniques et par la conception de batteries toujours plus performantes, rendus accessibles par la standardisation des protocoles et par l'utilisation d'un matériel peu couteux dans leur fabrication, les réseaux de capteurs sans fil constituent la solution technique idéale pour répondre à ces besoins. Ils sont ainsi pleinement intégrés au processus de surveillance des sites sensibles, à la gestion du trafic urbain, ou encore à l'étude de l'environnement naturel. Les capteurs comptent aussi de plus en plus d'applications médicales, ou bien appartenant au domaine militaire.

De tels usages ne peuvent être mis en œuvre sans garanties strictes en matière de sécurité. La confidentialité, l'authentification, l'intégrité des communications doivent être assurées. Ces propriétés constituent des enjeux essentiels et récurrents dans le domaine de l'informatique. Elles sont en général apportées par l'usage de protocoles cryptographiques : des mécanismes de chiffrement, de condensat et de signature des messages offrent des solutions efficaces.

Mais ces algorithmes ne sont pas toujours adaptés aux réseaux de capteurs sans fil, qui disposent de ressources matérielles très limitées. Pouvoir déployer un réseau de

plusieurs centaines de capteurs, dans un environnement parfois hostile, et sans forcément pouvoir les raccorder au réseau électrique implique en effet de réaliser certaines concessions : les capteurs disposent d'un matériel peu couteux, et leurs capacités de calcul, leur mémoire disponible, ainsi que la quantité d'énergie que peut leur fournir leur batterie sont relativement faibles. Ces contraintes n'ont cependant pas arrêté la conception de solutions adaptées pour garantir la confidentialité ou l'authentification des échanges dans ces réseaux.

C'est encore une autre propriété, propre elle aussi à la sécurité des réseaux, que nous avons étudiée dans cet ouvrage : la disponibilité des réseaux de capteurs sans fil. Autrement dit, leur capacité à prévenir, ou bien à détecter puis contrecarrer les attaques dites par « déni de service ». Ces attaques peuvent prendre de nombreuses formes, selon qu'elles viennent de l'intérieur ou de l'extérieur du réseau, selon qu'elles sont menées ou non à capacité matérielle égale ; ou encore selon le mode opératoire employé, selon la couche réseau visée. Pour chacune de ces formes connues, des solutions ont été proposées dans la littérature : changements de fréquence pour échapper au brouillage du canal, utilisation de « tunnels » virtuels, systèmes de détection d'intrusion, *et cætera*. Il est cependant essentiel de continuer à trouver de nouveaux mécanismes, ou d'améliorer ceux qui existent, afin de les rendre plus efficaces ou plus économes en ressources, tant pour obtenir des réseaux plus performants que pour rester à la hauteur face à de nouveaux types d'attaques.

2 Méthodes proposées

2.1 Des mécanismes…

Les travaux présentés dans cet ouvrage s'intéressent particulièrement à des solutions permettant de détecter des capteurs compromis au sein du réseau. Ces méthodes consistent à mettre en œuvre une surveillance du trafic échangé entre les nœuds d'un même cluster du réseau. Cette surveillance est réalisée par une sélection de capteurs du cluster, que l'on appelle « *cNodes* », et qui appliquent un ensemble de règles sur le trafic relevé dans leur voisinage. Si un capteur vient à enfreindre ces règles à de multiples reprises, par exemple en dépassant largement le seuil « autorisé » pour le débit d'envoi des paquets, alors le cluster head en est notifié, et peut être amené à exclure (virtuellement) le suspect du réseau.

Sur ces méthodes sont basés à la fois de nouveaux mécanismes et des modèles, tous destinés à améliorer le renouvellement du processus de sélection des *cNodes* (en particulier), et par là les capacités du réseau à contenir les attaques par déni de service (de façon plus générale).

Les trois mécanismes sont basés sur l'hypothèse suivante : attribuer le rôle de *cNode* à un nœud va automatiquement le pousser à consommer davantage d'énergie, et il est donc impératif de répartir le surplus engendré par cette charge entre les différents capteurs du cluster pour obtenir un équilibre correct, pour ce qui est de la consommation en énergie et donc de la durée de vie du réseau. Le risque, dans le cas contraire, est de voir un sous-ensemble restreint des capteurs tomber hors service longtemps avant tous les autres, laissant du même coup le cluster sans surveillance.

Peu de travaux existent à propos de la façon précise dont peuvent être désignées ces sentinelles. La première contribution consiste donc à introduire un mécanisme de

renouvellement simple à mettre en œuvre : une sélection aléatoire des *cNodes* qui peut être implémentée par la station de base du réseau (au détriment de l'aspect décentralisé du réseau), par les cluster heads, ou encore par les capteurs eux-mêmes, capables de s'auto-désigner aléatoirement. Cette méthode assure une bonne rotation des *cNodes*, et le processus aléatoire empêche un nœud compromis de l'utiliser à son avantage pour être désigné à chaque tour (et réduire ainsi les risques de détection de son attaque, ou bien pour « dénoncer » à tort des nœuds légitimes). Elle permet surtout de répartir le surplus de consommation en énergie que ne prendrait pas en compte une implémentation « statique » du système de surveillance, comme l'attestent les résultats numériques obtenus par simulation à l'aide du logiciel ns.

Pourtant, il est possible de pousser encore plus loin la recherche d'un équilibre optimal, toujours dans le but de répartir la charge le plus équitablement possible. Pour ce faire, nous proposons dans un deuxième temps une méthode qui, justement, repose sur l'énergie résiduelle des nœuds au moment du renouvellement : les capteurs possédant le plus d'énergie en réserve seront désignés *cNodes*. Les résultats de simulation montrent un écart-type très bas entre les nœuds sur la consommation énergétique, preuve que l'équilibre obtenu est très bon. Pourtant cette méthode a ses inconvénients : reposant sur un processus déterministe, elle introduit un risque quant à la couverture du cluster par les *cNodes*, menacée si les nœuds d'une certaine zone géographique sont amenés, par l'application qu'ils exécutent, à consommer davantage que leurs pairs. Le cluster head est donc responsable du contournement de ce problème, en veillant à la bonne couverture du cluster par les sentinelles. Mais il reste un second problème : un capteur compromis peut cette fois-ci jouer sur le processus de sélection, et annoncer une réserve d'énergie très élevée (mais factice) dans le but d'être systématiquement sélectionné. Pour parer cette faille potentielle, chaque *cNode* est entouré d'un ou de plusieurs « *vNodes* », chargés de l'interroger régulièrement sur son énergie résiduelle et de comparer ses réponses avec un modèle théorique de consommation, le tout afin de détecter les tentatives de fraude. Ce second rôle vient charger la méthode en complexité d'implémentation, ainsi qu'en volume de données de contrôle nécessaire : la consommation en énergie est effectivement mieux répartie, cependant elle est aussi plus élevée.

Aussi avons-nous été amenés à proposer une troisième contribution, toujours orientée à la fois vers l'équilibre de la consommation et vers la sécurité du processus. Ce troisième mécanisme de renouvellement est un processus d'élection démocratique, au cours duquel ce sont les nœuds du cluster qui votent auprès de leur cluster head afin de déterminer quels seront les *cNodes*. Tous les capteurs votent pour la première sélection, qui survient à la suite d'une période initiale où tous demeurent à l'écoute du trafic. Puis ce sont les *cNodes* d'une phase donnée qui votent pour leurs successeurs, au vu des conditions de trafic observées pendant leur « mandat ». Cette méthode permet de réutiliser les observations réalisées par les *cNodes* (dans le but de détecter un nœud compromis) lors du renouvellement de la sélection, sans surcharger le processus avec un nouveau type de nœuds. Elle permet aussi aux votants de tenir compte d'autres critères (index de connectivité des nœuds, puissance du signal reçu, score de réputation acquis avec le temps par exemple), ce qui n'est pas nécessairement notre objectif lorsque nous nous concentrons sur l'équilibre de la charge, mais qui peut représenter un avantage pour d'autres situations. Les valeurs obtenues lors des simulations, en comparaison avec les deux méthodes précédentes, sont bonnes, malgré une consommation énergétique très intense lors de la période initiale.

L'élection par les *cNodes* de cette dernière méthode implique un échange de données de contrôle légèrement plus important qu'une sélection aléatoire des nœuds,

c'est-à-dire une consommation globale très légèrement plus élevée (pour un meilleur équilibre). Le choix entre ces deux méthodes se fera donc suivant le réseau et l'application considérés, en fonction du besoin d'assurer en priorité la longévité maximale pour les « derniers capteurs » (méthode aléatoire) ou bien de conserver en service le plus longtemps possible un maximum de capteurs (méthode d'élection démocratique). La méthode de sélection selon l'énergie résiduelle ne viendra remplacer la méthode d'élection démocratique que dans de rares cas d'applications où les capteurs ne resteraient actifs que sur de courtes périodes, et où la réactivation du cluster entrainerait le besoin systématique de relancer une période initiale, avec la méthode d'élection démocratique, très couteuse en énergie.

2.2 ... et des modèles

En complément de ces mécanismes permettant de désigner les *cNodes* parmi les nœuds du cluster, différents modèles formels ont été établis. Le processus de sélection aléatoire des capteurs de surveillance a ainsi été représenté de trois façons différentes.

Les chaines de Markov à temps continu ont d'abord été utilisées pour représenter les transitions entre les différents états du système, et pour calculer la probabilité qu'un nœud donné soit compromis en fonction des mesures réalisées. Mais ce modèle suppose que la vérification du trafic par les *cNodes* est réalisée de façon ponctuelle, à intervalles aléatoires. Il s'agit d'une approximation trop large pour pouvoir adapter proprement cet outil au système modélisé.

Nous nous sommes donc tournés vers d'autres représentations, et avons modélisé chaque type de capteur, à la fois sous sa forme « statique » et dans un contexte de renouvellement dynamique, sous la forme de réseaux de Petri stochastiques généralisés étendus (comportant des transitions minutées). Une fois les capteurs représentés, nous avons pu formaliser une représentation de l'intégralité du cluster ainsi que du processus de renouvellement des *cNodes*.

Parmi les intérêts du modèle figure la possibilité de le réutiliser en combinaison avec un autre outil, la logique stochastique avec automates hybrides, dans le but d'en extraire par la suite des propriétés, d'en évaluer les performances ou la fiabilité, à l'aide d'outils de *model checking*. L'automate linéaire hybride associé au réseau de Petri, ainsi que des exemples de formules dans cette logique stochastique, sont fournis, et permettent de calculer la valeur de plusieurs variables relatives au système.

Pour information, le processus de sélection selon l'énergie résiduelle des capteurs a lui aussi fait l'objet d'une modélisation en vue d'appliquer des techniques de *model checking* [54], mais celle-ci n'entre pas dans le champ des travaux présentés dans cet ouvrage.

Tous ces modèles sont proposés dans le cadre de l'étude et de la validation des mécanismes introduits. Ils peuvent d'ailleurs être facilement adaptés à des configurations différentes de réseaux. Mais la validation *a posteriori* d'un processus n'est pas le seul apport que peuvent constituer les méthodes formelles : il est également intéressant de pouvoir bâtir une architecture, un protocole, sur un modèle établi au préalable. Dans cette optique, nous avons entrepris de modéliser les interactions entre un nœud compromis et les capteurs sains du réseau sous la forme de jeux quantitatifs. À la différence des travaux antérieurs, nous combinons des objectifs portant à la fois sur une valeur de gain (moyen) et sur une composante d'énergie. Ce modèle n'a pas encore permis d'apporter de nouveaux mécanismes de sécurité, mais il constitue une base intéressante d'étude, et permet de fournir de premiers résultats. Nous donnons ainsi les

spécifications des graphes de jeu, notamment au travers d'exemples détaillés, avant de nous intéresser à leur résolution : si la résolution de ces jeux constitue un problème indécidable dans le cas général, nous démontrons qu'il est possible de résoudre les problèmes de victoire et de crédit initial lorsque l'objectif à atteindre est défini sous forme de conjonctions de littéraux.

3 Perspectives pour les travaux futurs

Nous avons déjà mené une étude poussée sur les processus de renouvellement des *cNodes*, mais il reste néanmoins beaucoup de pistes de réflexion à explorer.

Varier les méthodes de détection et les réactions aux attaques Notre modèle de détection s'adapte bien à différents modèles d'attaques tant qu'il existe des règles permettant de traduire la signature des attaques en termes de trafic observé. Un angle d'approche concernerait l'étude et l'adoption, à la place, d'autres mécanismes (basés par exemple sur les anomalies de fonctionnement), dans le but de détecter de nouveaux schémas d'attaque. Et une fois la détection réalisée, il faudrait se pencher sur les réactions à adopter en réponse : y aurait-il, avec nos solutions, des éléments à mettre en œuvre à la place (ou bien en addition) de l'exclusion des capteurs compromis ? La création d'un nouveau canal de communication, que l'attaquant exclu ne pourrait rejoindre, est un exemple à envisager.

D'autres approches de lutte contre le déni de service Outre les méthodes de détection, c'est l'approche même, du point de vue de la sécurité, qui peut être étendue pour cibler d'autres types d'attaques par déni de service. La question du brouillage, par exemple, a été peu abordée : il pourrait être intéressant de s'appuyer en détail sur les spécifications d'un protocole choisi de couche de liaison de données, d'en trouver les faiblesses face aux tentatives de brouillage intelligent par exemple, et d'en dériver des mécanismes à inclure au niveau de la surveillance effectuée par les *cNodes*. Dans le même ordre d'idée, le travail sur la sécurité des algorithmes de clusterisation, les mécanismes de confiance, ou sur la résilience du routage face aux attaques, ne sont que quelques exemples de sujets brièvement étudiés au cours de cette thèse, mais qui mériteraient chacun d'être approfondis.

D'autres domaines de mise en œuvre En se recentrant sur les mécanismes décrits et implémentés sous forme de simulations, il serait possible d'étendre les cas étudiés à d'autres topologies de réseaux, à d'autres architectures que des systèmes clusterisés où tous les capteurs sont voisins directs d'un cluster head. Les différentes méthodes de renouvellement des *cNodes* proposées pourraient encore être complétées, soit par de nouveaux mécanismes qui rechercheraient des performances encore meilleures, soit dans un contexte radicalement différent. Par exemple, elles pourraient être transposées aux VANET, pour lesquels la mobilité est plus accentuée, qui possèdent une connectivité très différente entre les nœuds, mais dans lesquels le problème de l'énergie ne se pose (pratiquement) pas.

Des applications réelles : contraintes de qualité de service Toutes ces méthodes méritent également d'être étudiées dans le cadre d'applications plus spécifiques (en s'extrayant du cadre d'un comportement général). En fonction des scénarios déployés, les mécanismes de sélection devront peut-être subir de nouvelles adaptations. Imaginons par exemple une application qui introduirait une différenciation de services, avec plusieurs flux de nature distincte dans le réseau, et qui imposerait à l'architecture des contraintes de qualité de service : non seulement les contraintes risquent de se répercuter sur la mise en place de la surveillance, mais elles pourraient même, de par les nouvelles exigences vis-à-vis du bon fonctionnement des services, rendre possible de nouvelles attaques qu'il faudrait alors prévenir.

Mise en application Quoi qu'il en soit, les résultats numériques obtenus par simulation profiteraient logiquement de l'usage d'applications plus proches d'une mise en production réelle.

Réalisation d'une plate-forme opérationnelle Mais nous pourrions faire encore mieux : implémenter de tels scénarios sur un véritable réseau de capteurs, en usant par exemple du système d'exploitation TinyOS [126], ce qui nous permettrait d'obtenir des valeurs en meilleure adéquation avec une mise en production du système.

Validation des modèles proposés Il en va de même pour les modèles présentés, qui gagneraient d'une part à être mis en œuvre sur des configurations différentes, et d'une autre à être effectivement réutilisés avec les outils de *model checking* que nous avons mentionnés. Il serait particulièrement intéressant de mettre en place le renouvellement des *cNodes* sur un véritable cas d'application, sur un véritable réseau, et de modéliser celui-ci sous forme de réseaux de Petri pour vérifier si les variables formelles obtenues correspondent bien à la situation réelle.

Vers une modélisation toujours plus précise En fonction des résultats, il serait alors possible de rechercher l'amélioration des modèles obtenus, ou bien l'utilisation d'outils plus adaptés pour obtenir des modèles plus fidèles et plus efficaces.

Étude de nouveaux modèles Les automates temporels sont l'une des pistes envisageables à ce sujet. Les réseaux d'automates stochastiques en constituent une seconde : leur gestion efficace des évènements synchronisés pourrait leur donner un avantage conséquent, dans le contexte étudié, par rapport aux chaines de Markov, qui présentent des faiblesses sur ce point particulier.

Approfondissement des modèles développés Enfin, le modèle utilisant les jeux quantitatifs constitue une base de travail intéressante que nous aimerions affiner, dans le but d'améliorer le mécanisme de détection lui-même, par exemple en parvenant à exprimer des contraintes encore plus fortes qui forceraient les nœuds compromis à coopérer sous peine d'être immédiatement détectés.

Toutes ces approches sont autant de perspectives à étudier dans de futurs travaux, afin de concevoir, à partir des solutions apportées au cours de cette thèse, des mécanismes et des modèles toujours plus performants dans la lutte contre les attaques par déni de service dans les réseaux de capteurs sans fil.

Bibliographie

[1] Ameer Ahmed Abbasi et Mohamed Younis. « A Survey on Clustering Algorithms for Wireless Sensor Networks ». In : *Computer Communications* 30.14–15 (oct. 2007), p. 2826–2841 (cf. p. 21).

[2] Abdelrahman Abuarqoub, Fayez Al-Fayez, Tariq Alsboui, Mohammad Hammoudeh et Andre Nisbet. « Simulation Issues in Wireless Sensor Networks : A Survey ». In : *Proceedings of the 6th International Conference on Sensor Technologies and Applications (SENSORCOMM'12)*. Rome, Italie, août 2012, p. 222–228 (cf. p. 23).

[3] Afrand Agah et Sajal K. Das. « Preventing DoS Attacks in Wireless Sensor Networks. A Repeated Game Theory Approach ». In : *International Journal of Network Security* 5.2 (sept. 2007), p. 145–153 (cf. p. 131).

[4] Nadeem Ahmed, Salil S. Kanhere et Sanjay Jha. « The Holes Problem in Wireless Sensor Networks : A Survey ». In : *ACM SIGMOBILE Mobile Computing and Communications Review* 9.2 (avr. 2005), p. 4–18 (cf. p. 41).

[5] Marco Ajmone Marsan, Gianfranco Balbo, Giuseppe Conte, Susanna Donatelli et Giuliana Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley & Sons, nov. 1995 (cf. p. 84).

[6] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam et E. Cayirci. « Wireless Sensor Networks : A Survey ». In : *Computer Networks* 38.4 (mar. 2002), p. 393–422 (cf. p. 10, 11).

[7] Sahabul Alam et Debashis De. « Analysis of Security Threats in Wireless Sensor Network ». In : *International Journal of Wireless and Mobile Networks* 6.2 (avr. 2014), p. 35–46 (cf. p. 26, 35, 40–45).

[8] Giuseppe Anastasi, Marco Conti, Mario di Francesco et Andrea Passarella. « Energy Conservation in Wireless Sensor Networks : A Survey ». In : *Ad Hoc Networks* 7.3 (mai 2009), p. 537–568 (cf. p. 16, 17).

[9] Laleh Arshadi et Amir Hossein Jahangir. « Entropy Based SYN Flooding Detection ». In : *Proceedings of the 36th Annual IEEE Conference on Local Computer Networks (LCN'11)*. Bonn, Allemagne, oct. 2011, p. 139–142 (cf. p. 44).

[10] Paolo Ballarini, Hilal Djafri, Marie Duflot, Serge Haddad et Nihal Pekergin. « COSMOS : A Statistical Model Checker for the Hybrid Automata Stochatic Logic ». In : *Proceedings of the 8th International Conference on Quantitative Evaluation of Systems (QEST'11)*. Aachen, Allemagne, sept. 2011, p. 143–144 (cf. p. 93).

[11] Paolo Ballarini, Hilal Djafri, Marie Duflot, Serge Haddad et Nihal Pekergin. « HASL : An Expressive Language for Statistical Verification of Stochastic Models ». In : *Proceedings of the 5th International ICST Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS'11)*. Cachan, France, mai 2011, p. 306–315 (cf. p. 92).

[12] Paolo Ballarini, Lynda Mokdad et **Quentin Monnet**. « Modeling Tools for Detecting DoS Attacks in WSNs ». In : *Security and Communication Networks* 6.4 (avr. 2013), p. 420–436 (cf. p. 70).

[13] Paolo Baronti, Prashant Pillai, Vince W.C. Chook, Stefano Chessa, Alberto Gotta et Y. Fun Hu. « Wireless Sensor Networks : A Survey on the State of the Art and the 802.15.4 and ZigBee Standards ». In : *Computer Communications* 30.7 (mai 2007), p. 1655–1695 (cf. p. 16).

[14] Harjot Bawa, Parminder Singh et Rakesh Kumar. « An Efficient Novel Key Management Scheme for Enhancing User Authentication in a WSN ». In : *International Journal of Computer Network and Information Security* 5.1 (jan. 2013), p. 56–64 (cf. p. 30, 33).

[15] Gerd Behrmann, Alexandre David et Kim G. Larsen. « A Tutorial on Uppaal ». In : *Formal Methods for the Design of Real-Time Systems*. T. 3185. Springer, 2004, p. 200–236 (cf. p. 104).

[16] Soufiene Ben Othman, Abdullah Ali Bahattab, Abdelbasset Trad et Habib Youssef. « Confidentiality and Integrity for Data Aggregation in WSN Using Homomorphic Encryption ». In : *Wireless Personal Communications* (sept. 2014) (cf. p. 28).

[17] Jalel Ben-Othman et Lynda Mokdad. « Enhancing Data Security in Ad Hoc Networks Based on Multipath Routing ». In : *Journal of Parallel and Distributed Computing* 70.3 (mar. 2010), p. 309–316 (cf. p. 28).

[18] **Quentin Monnet** et Lynda Mokdad. « DoS Detection in WSNs : Energy-Efficient Designs and Modeling Tools for Choosing Monitoring Nodes ». In : *Modeling and Simulation of Computer Networks and Systems*. Sous la dir. de Mohammad S. Obaidat, Faouzi Zaraia et Petros Nicopolitidis. Elsevier, avr. 2015. Chap. 28, p. 795–840 (cf. p. 99).

[19] **Quentin Monnet**, Lynda Mokdad et Jalel Ben-Othman. « Data Protection in Multipaths WSNs ». In : *Proceedings of the 18th IEEE Symposium on Computers and Communications (ISCC'13)*. Split, Croatie, juil. 2013 (cf. p. 28, 50).

[20] **Quentin Monnet**, Lynda Mokdad et Jalel Ben-Othman. « Energy-Balancing Method to Detect Denial of Service Attacks in Wireless Sensor Networks ». In : *Proceedings of the IEEE International Conference on Communications (ICC'14)*. Sydney, NSW, Australie, juin 2014 (cf. p. 99).

[21] Nikita Borisov, Ian Goldberg et David Wagner. « Intercepting Mobile Communications : The Insecurity of 802.11 ». In : *Proceedings of the ACM 7th Annual International Conference on Mobile Computing and Networking (SIGMOBILE)*. Rome, Italie, juil. 2001, p. 180–189 (cf. p. 29).

[22] Ismail Butun, Salvatore D. Morgera et Ravi Sankar. « A Survey of Intrusion Detection Systems in Wireless Sensor Networks ». In : *IEEE Communications Surveys and Tutorials* 16.1 (mai 2013), p. 266–282 (cf. p. 11, 22, 53–55, 57, 59, 60).

[23] Mario Čagalj, Srdjan Čapkun et Jean-Pierre Hubaux. « Wormhole-Based Anti-Jamming Techniques in Sensor Networks ». In : *IEEE Transactions on Mobils Computing* 6.1 (jan. 2007), p. 100–114 (cf. p. 61).

[24] CEREMA – Centre d'études et d'expertise sur les risques, l'environnement, la mobilité et l'aménagement. *Les transports intelligents*. Page consultée le 14 février 2015. url : http://www.transport-intelligent.net/technologies/capteurs-77/ (cf. p. 10, 17).

[25] Krishnendu Chatterjee, Tom Henzinger et Marcin Jurdziński. « Mean-Payoff Parity Games ». In : *Proceedings of the 20th International Symposium on Logic In Computer Science (LICS'05)*. Juin 2005 (cf. p. 131).

[26] Krishnendu Chatterjee, Mickael Randour et Jean-François Raskin. « Strategy Synthesis for Multi-Dimensional Quantitative Objectives ». In : *Proceedings of the 23rd International Conference on Concurrency Theory (CONCUR'12)*. Sous la dir. de Maciej Koutny et Irek Ulidowski. T. 7454. Lecture Notes in Computer Science. Springer, sept. 2012, p. 115–131 (cf. p. 131).

[27] Naveen Chawla et Ashish Jasuja. « Algorithm for Optimizing First Node Die (FND) Time in LEACH Protocol ». In : *International Journal of Current Engineering and Technology* 4.4 (août 2014), p. 2748–2750 (cf. p. 21).

[28] Youngho Cho, Gang Qu et Yuanming Wu. « Insider Threats Against Trust Mechanism with Watchdog and Defending Approaches in Wireless Sensor Networks ». In : *Proceedings of the 2012 IEEE Symposium on Security and Privacy Workshops (SPW'12)*). San Francisco, CA, États-Unis, mai 2012, p. 134–141 (cf. p. 59).

[29] Pau Closas, Alba Pagès-Zamora et Juan A. Fernández-Rubio. « A Game Theoretical Algorithm for Joint Power and Topology Control in Distributed WSN ». In : *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'09)*. Athènes, Grèce, avr. 2009, p. 462–468 (cf. p. 130).

[30] Roberto Doriguzzi Corin, Giovanni Russello et Elio Salvadori. « TinyKey : A Light-Weight Architecture for Wireless Sensor Networks Securing Real-World Applications ». In : *Proceedings of the 8th International Conference on Wireless On-Demand Network Systems and Services (WONS'11)*. Bardonecchia, Italie, jan. 2011, p. 68–75 (cf. p. 31).

[31] Garth V. Crosby, Niki Pissinou et James Gadze. « A Framework for Trust-Based Cluster Head Election in Wireless Sensor Networks ». In : *Proceedings of the 2nd IEEE Workshop on Dependability and Security in Sensor Networks and Systems (DSSNS'06)*. Columbia, MD, États-Unis, avr. 2006 (cf. p. 62).

[32] Benjamin J. Culpepper et H. Chris Tseng. « Sinkhole Intrusion Indicators in DSR MANETs ». In : *Proceedings of the 1st International on Broadband Networks BroadNets'04)*. San José, CA, États-Unis, oct. 2004, p. 681–688 (cf. p. 41).

[33] Ahmad Yusri Dak, Saadiah Yahya et Murizah Kassim. « A Literature Survey on Security Challenges in VANETs ». In : *International Journal of Computer Theory and Engineering* 4.6 (déc. 2012), p. 1007–1010 (cf. p. 22, 26).

[34] Hongmei Deng, Wei Li et Dharma P. Agrawal. « Routing Security in Wireless Ad Hoc Networks ». In : *IEEE Communications Magazine* 40.10 (oct. 2002), p. 70–75 (cf. p. 50).

[35] Abdoulaye Diop, Yue Qi, Qin Wang et Shariq Hussain. « An Advanced Survey on Secure Energy-Efficient Hierarchical Routing Protocols in Wireless Sensor Networks ». In : *International Journal of Computer Science Issues* 10.1 (jan. 2013), p. 490–500 (cf. p. 21, 22).

[36] K. Drira, H. Seba et H. Kheddouci. « ECGK : An Efficient Clustering Scheme for Group Key Management in MANETs ». In : *Computer Communications* 33.9 (juin 2010), p. 1094–1107 (cf. p. 27, 33).

[37] Nicolas Falliere, Liam O. Murchu et Eric Chien. *W32.Stuxnet Dossier*. Symantec. Fév. 2011 (cf. p. 49).

[38] Najma Farooq, Irwa Zahoor et Sandip Mandal. « Recovering from In-Band Wormhole Based Denial of Service in Wireless Sensor Networks ». In : *International Journal of Current Engineering and Technology* 4.3 (juin 2014), p. 1604–1607 (cf. p. 42).

[39] M. Carmen Fernández-Gago, Rodrigo Román et Javier Lopez. « A Survey on the Applicability of Trust Management Systems for Wireless Sensor Networks ». In : *Proceedings of the 3rd International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing (SECPerU'07)*. Istanbul, Turquie, juil. 2007, p. 25–30 (cf. p. 63).

[40] Said Fouchal, **Quentin Monnet**, Djamel Mansouri, Lynda Mokdad et Malika Ioualalen. « A new Clustering Algorithm for Wireless Sensor Networks ». In : *Proceedings of the 17th IEEE Symposium on Computers and Communications (ISCC'12)*. Nevşehir, Turquie, juil. 2012 (cf. p. 21).

[41] Said Fouchal et Ivan Lavallée. « Fast and Flexible Unsupervised Clustering Algorithm Based on Ultrametric Properties ». In : *Proceedings of the 7th ACM Symposium on QoS and Security for Wireless and Mobile Networks (Q2SWinet'11)*. Miami, FL, États-Unis, nov. 2011 (cf. p. 21).

[42] Aurélien Francillon et Claude Castelluccia. « Code Injection Attacks on Harvard-Architecture Devices ». In : *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS'08)*. Alexandria, VA, États-Unis, oct. 2008, p. 15–26 (cf. p. 51).

[43] Chinnu Mary George et Manoj Kumar. « Cluster Based Location Privacy in Wireless Sensor Networks Against a Universal Adversary ». In : *Proceedings of the International Conference on Information Communication and Embedded Systems (ICICES'13)*. Chennai, Inde, fév. 2013, p. 288–293 (cf. p. 44, 50).

[44] Amrita Ghosal et Sipra DasBit. « A Lightweight Security Scheme for Query Processing in Clustered Wireless Sensor Networks ». In : *Computers and Electrical Engineering* (avr. 2014) (cf. p. 30).

[45] Zheng Gong, Svetla Nikova et YeeWei Law. « KLEIN : A New Family of Lightweight Block Ciphers ». In : *RFID. Security and Privacy*. Sous la dir. d'Ari Juels et Christof Paar. T. 7055. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, p. 1–18 (cf. p. 16, 27).

[46] Erich Grädel, Wolfgang Thomas et Thomas Wilke. *Automata, Logics, and Infinite Games*. Springer, oct. 2002 (cf. p. 133).

[47] Qijun Gu, Christopher Ferguson et Rizwan Noorani. « A Study of Self Propagating Mal-Packets in Sensor Networks : Attacks and Defenses ». In : *Computers & Security* 30.1 (jan. 2011), p. 13–27 (cf. p. 51).

[48] Malek Guechari, Lynda Mokdad et Sovanna Tan. « Dynamic Solution for Detecting Denial of Service Attacks in Wireless Sensor Networks ». In : *Proceedings of the 2012 IEEE International Conference on Communications (ICC'12)*. Ottawa, Canada, juin 2012 (cf. p. 58, 73).

[49] Ping Guo, Jin Wang, Jiezhong Zhu et Yaping Cheng. « Authentication Mechanism on Wireless Sensor Networks : A Survey ». In : *Proceedings of the 2nd International Conference on Information Technology and Computer Science (ITCS'13)*. T. 25. Beijing, Chine, juil. 2013, p. 425–431 (cf. p. 30).

[50] Ping Guo, Jin Wang, JieZhong Zhu, YaPing Cheng et Jeong-Uk Kim. « Construction of Trusted Wireless Sensor Networks with Lightweight Bilateral Authentication ». In : *International Journal of Security and Its Applications* 7.5 (sept. 2013), p. 225–236 (cf. p. 30).

[51] Tasneem Halim et Md. Rafiqul Islam. « A Study on the Security Issues in WSN ». In : *International Journal of Computer Applications* 53.1 (sept. 2012), p. 26–32 (cf. p. 30).

[52] Sina Hamedheidari et Reza Rafeh. « A Novel Agent-Based Approach to Detect Sinkhole Attacks in Wireless Sensor Networks ». In : *Computers and Security* 37 (sept. 2013), p. 1–14 (cf. p. 22, 59).

[53] Ali Hamieh. « La sécurité dans les réseaux sans fil Ad Hoc : les attaques jamming et les nœuds greedy ». PhD thesis. Versailles, France : University of Versailles, 2012 (cf. p. 130).

[54] Youcef Hammal, **Quentin Monnet**, Lynda Mokdad, Jalel Ben-Othman et Abdelkarim Abdelli. « Timed Automata Based Modeling and Verification of Denial of Service Attacks in Wireless Sensor Networks ». In : *Studia Informatica Universalis* 12.1 (2014), p. 1–46 (cf. p. 104, 152).

[55] M. J. Handy, M. Haase et D. Timmerman. « Low Energy Adaptive Clustering Hierarchy with Deterministic Cluster-Head Selection ». In : *Proceedings of the 4th IEEE International Workshop on Mobile and Wireless Communications Networks*. Stockholm, Suède, 2002, p. 368–372 (cf. p. 21).

[56] Anne-Marie Hegland, Eli Winjum, Stig F. Mjølsnes, Chunming Rong, Øivind Kure et Pål Spilling. « A Survey of Key Management in Ad Hoc Networks ». In : *IEEE Communications Surveys* 8.3 (mar. 2006), p. 48–66 (cf. p. 30, 33).

[57] Wendi Rabiner Heinzelman, Anantha Chandrakasan et Hari Balakrishnan. « Energy-Efficient Communication Protocol for Wireless Microsensor Networks ». In : *Proceedings of the IEEE 33rd Hawaii International Conference on System Sciences*. T. 8. Maui, HI, États-Unis, jan. 2000 (cf. p. 19).

[58] Jun-Won Ho. « Distributed Detection of Node Capture Attacks in Wireless Networks ». In : *Smart Wireless Sensor Networks*. Sous la dir. d'Hoang Duc Chinh et Yen Kheng Tan. InTech, déc. 2010. Chap. 20, p. 345–360 (cf. p. 33, 50).

[59] Meng-Yen Hsieh, Yueh-Min Huang et Han-Chieh Chao. « Adaptive Security Design with Malicious Node Detection in Cluster-Based Sensor Networks ». In : *Computer Communications* 30 (sept. 2007), p. 2385–2400 (cf. p. 58).

[60] Jyh-Ming Huang, Shun-Bo Yang et Chan-Ling Dai. « An Efficient Key Management Scheme for Data-Centric Storage Wireless Sensor Networks ». In : *Proceedings of the 3rd International Conference on Electronic Engineering and Computer Science (EECS'13)*. T. 4. Hong Kong, Chine, déc. 2013, p. 25–31 (cf. p. 32).

[61] *IEEE Standard 802.11™ (révision de 2012) Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. IEEE Standard for Information technology, Telecommunications and information exchange between systems, Local and metropolitan area networks (cf. p. 39).

[62] Internet Engineering Task Force (IETF). *Internet-Draft : Survey of Security Hardening Methods for Transmission Control Protocol (TCP) Implementations*. Brouillon expiré en septembre 2012. Mar. 2012 (cf. p. 43).

[63] Internet Engineering Task Force (IETF). *RFC 3561 : Ad Hoc On-Demand Distance Vector (AODV) Routing*. Statut expérimental. Juil. 2003 (cf. p. 14).

[64] Internet Engineering Task Force (IETF). *RFC 3626 : Optimized Link State Routing Protocol (OLSR)*. Statut expérimental. Oct. 2003 (cf. p. 14).

[65] Internet Engineering Task Force (IETF). *RFC 4728 : The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4*. Statut expérimental. Fév. 2007 (cf. p. 14).

[66] Muhammad Hasan Islam, Kamran Nadeem et Shoab A. Khan. « Optimal Sensor Placement for Detection Against Distributed Denial of Service Attacks ». In : *Proceedings of the 2009 International Conference on Advanced Computer Control (ICACC'09)*. Singapour, jan. 2009, p. 675–679 (cf. p. 58).

[67] Nafaa Jabeura, Nabil Sahlib et Ijaz Muhammad Khan. « Survey on Sensor Holes : A Cause-Effect-Solution Perspective ». In : *Proceedings of the 8th International Symposium on Intelligent Systems Techniques for Ad Hoc and Wireless Sensor Networks (IST-AWSN'13)*. T. 19. Halifax, NS, Canada, juin 2013, p. 1074–1080 (cf. p. 41).

[68] Raj Jain. *The art of Computer Performance Analysis*. Wiley, avr. 1991 (cf. p. 70, 71).

[69] D. C. Jinwala, Dhiren R. Patel et K. S. Das Gupta. « A Survey of the Security Issues in Wireless Sensor Networks ». In : *A. D. Patel Institute of Technology Journal of Engineering* (2006) (cf. p. 36).

[70] Zdravko Karakehayov. « Using REWARD to Detect Team Black-Hole Attacks in Wireless Sensor Networks ». In : *Proceedings of the 1st Workshop on Real-World Wireless Sensor Networks (REALWSN'05)*. Stockholm, Suède, juin 2005, p. 75–79 (cf. p. 40, 59).

[71] Chris Karlof, Naveen Sastry et David Wagner. « TinySec : A Link Layer Security Architecture for Wireless Sensor Networks ». In : *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*. Baltimore, MD, États-Unis, nov. 2004, p. 162–175 (cf. p. 31).

[72] A. Babu Karuppiah et S. Rajaram. « Energy Efficient Encryption Algorithm for Wireless Sensor Network ». In : *International Journal of Engineering Research and Technology* 1.3 (mai 2012). Whitepaper (cf. p. 27).

[73] Leonidas Kazatzopoulos, Costas Delakouridis et Christos Anagnostopoulos. « WSN Location Privacy Scheme Enhancement through Epidemical Information Dissemination ». In : *International Journal of Communication Networks and Information Security* 6.2 (août 2014), p. 162–167 (cf. p. 9, 50).

[74] Asif Khan, Israfil Tamim, Emdad Ahmed et Muhammad Abdul Awal. « Multiple Parameter Based Clustering (MPC) : Prospective Analysis for Effective Clustering in Wireless Sensor Network (WSN) Using k-means Algorithm ». In : *Wireless Sensor Network* 4.1 (jan. 2012), p. 18–24 (cf. p. 21).

[75] Kashif Kifayat, Madjid Merabti, Qi Shi et David Llewellyn-Jones. « An Efficient Multi-Parameter Group Leader Selection Scheme for Wireless Sensor Networks ». In : *Proceedings of the 1st International Conference on Network and Service Security (N2S'09)*. Paris, France, juin 2009, p. 110–114 (cf. p. 21).

[76] Saru Kumari, Muhammad Khurram Khan et Mohammed Atiquzzaman. « User Authentication Schemes for Wireless Sensor Networks : A Review ». In : *Ad Hoc Networks* 27 (avr. 2015), p. 159–194 (cf. p. 29).

[77] Gu Hsin Lai et Chia-Mei Chen. « Detecting Denial of Service Attacks in Sensor Networks ». In : *Journal of Computers* 4.18 (jan. 2008) (cf. p. 58, 60, 68).

[78] Yee Wei Law, Jeroen Doumen et Pieter Hartel. « Survey and Benchmark of Block Ciphers for Wireless Sensor Networks ». In : *ACM Transactions on Sensor Networks* 2.1 (fév. 2006), p. 65–93 (cf. p. 16).

[79] Bai Li et Lynn Batten. « Using Mobile Agents to Recover from Node and Database Compromise in Path-Based DoS Attacks in Wireless Sensor Networks ». In : *Journal of Network and Computer Applications* 32.2 (mar. 2009), p. 337–387 (cf. p. 60).

[80] Zhijun Li et Guang Gong. « A Survey on Security in Wireless Sensor Networks ». In : *Special English Edition of Journal of Korean Institute of Information Security* 18.6 (déc. 2008), p. 233–248 (cf. p. 27).

[81] Vincent Liu, Aaron Parks, Vamsi Talla, Shyamnath Gollakota, David Wetherall et Joshua R. Smith. « Ambient Backscatter : Wireless Communication out of Thin Air ». In : *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*. Hong Kong, Chine, août 2013, p. 39–50 (cf. p. 17).

[82] Mark Luk, Ghita Mezzour, Adrian Perrig et Virgil Gligor. « MiniSec : A Secure Sensor Network Communication Architecture ». In : *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (ISPN'06)*. Cambridge, MA, États-Unis, avr. 2007, p. 479–488 (cf. p. 31).

[83] Jun Luo, Panagiotis Papadimitratos et Jean-Pierre Hubaux. « GossiCrypt : Wireless Sensor Network Data Confidentiality Against Parasitic Adversaries ». In : *Proceedings of the 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'08)*. San Francisco, CA, États-Unis, juin 2008, p. 441–450 (cf. p. 28).

[84] Xi Luo, Yi-Ying Zhang, Wen-Cheng Yang et Myong-Soon Park. « Prevention of DoS Attacks Based on Light Weight Dynamic Key Mechanism in Hierarchical Wireless Sensor Networks ». In : *Proceedings of the 2nd International Conference on Future Generation Communication and Networking (FGCN'08)*. Sanya, Hainan, Chine, déc. 2008, p. 309–312 (cf. p. 70).

[85] Renita Machado et Sirin Tekinay. « A Survey of Game-Theoric Approaches in Wireless Sensor Networks ». In : *Computer Networks* 52.16 (nov. 2008), p. 3047–3061 (cf. p. 130).

[86] G. Mahalakshmi et P. Subathra. « A Survey on Prevention Approaches for Denial of Sleep Attacks in Wireless Networks ». In : *Journal of Emerging Technologies in Web Intelligence* 6.1 (fév. 2014), p. 106–110 (cf. p. 39, 44, 59).

[87] Yassine Maleh et Abdellah Ezzati. « A Review of Security Attacks and Intrusion Detection Schemes in Wireless Sensor Networks ». In : *International Journal of Wireless & Mobile Networks* 5.6 (déc. 2013) (cf. p. 59).

[88] Kirk Martinez, Paritosh Padhy, Alistair Riddoch, Royan Ong et Jane Hart. « Glacial Environment Monitoring using Sensor Networks ». In : *Proceedings of the 1st Workshop on Real-World Wireless Sensor Networks (REALWSN'05)*. Stockholm, Suède, juin 2005, p. 9–13 (cf. p. 9).

[89] Marvin L. Minsky. *Computation : Finite and Infinite Machines*. Upper Saddle River, NJ, États-Unis : Prentice-Hall, Inc., 1967 (cf. p. 139).

[90] Sudip Misra, P. Venkata Krishna, Kiran Isaac Abraham, Navin Sasikumar et S. Fredun. « An Adaptive Learning Routing Protocol for the Prevention of Distributed Denial of Service Attacks in Wireless Mesh Networks ». In : *Computer and Mathematics with Applications* 60.2 (2010), p. 294–306 (cf. p. 58).

[91] Maryam Mohi, Ali Movaghar et Pooya Moradian Zadeh. « A Bayesian Game Approach for Preventing DoS Attacks in Wireless Sensor Networks ». In : *Proceedings of the 2009 International Conference on Communications and Mobile Computing (CMC'09)*. Kunming, Yunnan, Chine, fév. 2009 (cf. p. 130).

[92] Mohammad Momani et Subhash Challa. « Survey of Trust Models in Different Network Domains ». In : *International Journal of Ad Hoc, Sensor and Ubiquitous Computing* 1.3 (sept. 2010), p. 1–19 (cf. p. 63).

[93] National Institute of Standards and Technology (NIST). *Federal Information Processing Standards Publication (FIPS) 197 : Advanced Encryption Standard (AES)*. Nov. 2001 (cf. p. 27).

[94] James Newsome, Elaine Shi, Dawn Song et Adrian Perrig. « The Sybil Attack in Sensor Networks : Analysis & Defenses ». In : *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN'04)*. Berkeley, CA, États-Unis, avr. 2004, p. 259–268 (cf. p. 43).

[95] Leonardo B. Oliveira, Adrian Ferreira, Marco A. Vilaça, Hao Chi Wong, Marshall Bern, Ricardo Dahab et Antonio A. F. Loureiro. « SecLEACH — On the Security of Clustered Sensor Networks ». In : *Signal Processing* 87.12 (déc. 2007), p. 2882–2895 (cf. p. 21, 33).

[96] Suat Ozdemir et Yang Xiao. « Secure Data Aggregation in Wireless Sensor Networks : A Comprehensive Overview ». In : *Computer Networks* 53.12 (août 2009), p. 2022–2037 (cf. p. 27, 36).

[97] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994 (cf. p. 146).

[98] Konstantinos Pelechrinis et Marios Iliofotou. « Denial of Service Attacks in Wireless Networks : The Case of Jammers ». In : *IEEE Communications Surveys and Tutorials* 13.2 (2011), p. 245–257 (cf. p. 37–40, 61).

[99] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler et J. D. Tygar. « SPINS : Security Protocols for Sensor Networks ». In : *Wireless Networks* 8.5 (sept. 2002), p. 521–534 (cf. p. 30).

[100] Krzysztof Piotrowski, Peter Langendoerfer et Steffen Peter. « How Public Key Cryptography Influences Wireless Sensor Node Lifetime ». In : *Proceedings of the 4th ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'06)*. Alexandria, VA, États-Unis, oct. 2006, p. 169–176 (cf. p. 16, 23, 27).

[101] D. Rakhmatov et S. Vrudhula. « An Analytical High-Level Battery Model for Use in Energy Management of Portable Electronic Systems ». In : *Proceedings of the International Conference on Computer Aided Design (ICCAD'01)*. San José, CA, États-Unis, nov. 2001, p. 488–493 (cf. p. 100).

[102] Vishal Rathod et Mrudang Mehta. « Security in Wireless Sensor Network : A Survey ». In : *Ganpat University Journal of Engineering and Technology* 1.1 (jan. 2011), p. 35–44 (cf. p. 34, 45).

[103] B. Brahma Reddy et K. Kishan Rao. « A Modified Clustering for LEACH Algorithm in WSN ». In : *International Journal of Advanced Computer Science and Applications* 4.5 (mai 2013), p. 79–83 (cf. p. 21).

[104] Yenumula B. Reddy. « A Game Theory Approach to Detect Malicious Nodes in Wireless Sensor Networks ». In : *Proceedings of the 3rd International Conference on Sensor Technologies and Applications (SENSORCOMM'09)*. Athènes, Grèce, juin 2009, p. 462–468 (cf. p. 131).

[105] Mohammad Reza Rohbanian, Mohammad Rafi Kharazmi, Alireza Keshavarz-Haddad et Manije Keshtgary. « Watchdog-LEACH : A New Method Based on LEACH Protocol to Secure Clustered Wireless Sensor Networks ». In : *Advances in Computer Science : An International Journal* 2.3 (juil. 2013), p. 105–117 (cf. p. 57).

[106] Wissam Sammouri, Étienne Côme, Latifa Oukhellou et Patrice Aknin. « Mining Floating Train Data Sequences for Temporal Association Rules within a Predictive Maintenance Framework ». In : *Advances in Data Mining. Applications and Theoretical Aspects*. Sous la dir. de Petra Perner. T. 7987. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, p. 112–126 (cf. p. 9).

[107] Kang Ho-Seok, Kim Sung-Ryula et Kim Pankoo. « Traffic Deflection Method for DoS Attack Defense using a Location-Based Routing Protocol in the Sensor Network ». In : *Computer Science and Information Systems* 10.2 (2013), p. 685–701 (cf. p. 62).

[108] H. Shafiei, A. Khonsari, H. Derakhshi et P. Mousavi. « Detection and Mitigation of Sinkhole Attacks in Wireless Sensor Networks ». In : *Journal of Computer and System Sciences* 80.3 (mai 2014), p. 644–653 (cf. p. 59).

[109] Vartika Shah et Sanjiv Sharma. « A Review of Existing Security Frameworks and Encryption Methods for Wireless Sensor Networks ». In : *International Journal of Innovations & Advancement in Computer Science* 3.2 (avr. 2014), p. 90–98 (cf. p. 30, 33).

[110] Vartika Shah et Sanjiv Sharma. « Evaluation of Encryption Method of SNEP for Implementing Security in Wireless Sensor Network using SPINS Framework ». In : *International Journal of Grid Distribution Computing* 7.5 (2014), p. 43–52 (cf. p. 31).

[111] Adi Shamir. « How to Share a Secret ». In : *Communications of the ACM* 22.11 (nov. 1979), p. 612–613 (cf. p. 28).

[112] Min Shao, Sencun Zhu, Wensheng Zhang, Guohong Cao et Yi Yang. « pDCS : Security and Privacy Support for Data-Centric Sensor Networks ». In : *IEEE Transactions on Mobile Computing* 8.8 (août 2009), p. 1023–1038 (cf. p. 31).

[113] Suraj Sharma et Sanjay Kumar Jena. « A Survey on Secure Hierarchical Routing Protocols in Wireless Sensor Networks ». In : *Proceedings of the 2011 International Conference on Communication, Computing and Security (ICCCS'11)*. Rourkela, Inde, fév. 2011, p. 146–151 (cf. p. 53).

[114] Hai-Yan Shi, Wan-Liang Wang, Ngai-Ming Kwok et Sheng-Yong Chen. « Game Theory for Wireless Sensor Networks : A Survey ». In : *Sensors* 12.7 (juil. 2012), p. 9055–9097 (cf. p. 130).

[115] Marcos A. Simplicio Jr, Bruno T. de Oliveira, Paulo S. L. M. Barreto, Cintia B. Margi, Tereza C. M. B. Carvalho et Mats Naslund. « Comparison of Authenticated-Encryption Schemes in Wireless Sensor Networks ». In : *Proceedings of the 36th Annual IEEE Conference on Local Computer Networks*. Bonn, Allemagne, oct. 2011, p. 454–461 (cf. p. 27).

[116] Shio Kumar Singh, M. P. Singh et D. K. Singh. « A Survey on Network Security and Attack Defense Mechanism for Wireless Sensor Networks ». In : *International Journal of Computer Trends and Technology* (mai 2011) (cf. p. 45).

[117] P. Sivakumar, K. Amirthavalli et M. Senthil. « Power Conservation and Security Enhancement in Wireless Sensor Networks : A Priority Based Approach ». In : *International Journal of Distributed Sensor Networks* (mai 2014) (cf. p. 17).

[118] Ju-Hyung Son, Haiyun Luo et Seung-Woo Seo. « Denial of Service Attack-Resistant Flooding Authentication in Wireless Sensor Networks ». In : *Computer Communications* 33.13 (août 2010), p. 1531–1542 (cf. p. 33).

[119] Eliana Stavrou et Andreas Pitsillides. « A Survey on Secure Multipath Routing Protocols in WSNs ». In : *Computer Networks* 54.13 (sept. 2010), p. 2215–2238 (cf. p. 23).

[120] Fangmin Sun, Zhan Zhao, Zhen Fang, Lidong Du, Zhihong Xu et Diliang Chen. « A Review of Attacks and Security Protocols for Wireless Sensor Networks ». In : *International Journal of Networks* 9.5 (mai 2014), p. 1103–1113 (cf. p. 10, 26, 35–37, 42, 44).

[121] Andrew S. Tanenbaum et David J. Wetherall. *Computer Networks, 5/E*. Prentice Hall, sept. 2010 (cf. p. 11–14, 38).

[122] Li Tao, Eugenio Cinquanta, Daniele Chiappe, Carlo Grazianetti, Marco Fanciulli, Madan Dubey, Alessandro Molle et Deji Akinwande. « Silicene Field-Effect Transistors Operating at Room Temperature ». In : *Nature Technology* (fév. 2015) (cf. p. 16).

[123] *The Network Simulator – ns-2*. url : https://ant.isi.edu/nsnam/index.php (cf. p. 74).

[124] *The Network Simulator – ns-3*. url : https://www.nsnam.org (cf. p. 74).

[125] Fabrice Theoleyre et Fabrice Valois. « VSR : A Routing Protocol Based on a Structure of Self-Organization ». In : *Studia Informatica Universalis* 6.1 (2008), p. 40–69 (cf. p. 21, 22).

[126] *TinyOS*. URL : http://www.tinyos.net/ (cf. p. 154).

[127] Llanos Tobarra, Diego Cazorla, Fernando Cuartero, Gregorio Díaz et Emilia Cambronero. « Model Checking Wireless Sensor Network Security Protocols : TinySec + LEAP + TinyPK ». In : *Telecommunication Systems* 40.3–4 (avr. 2009), p. 91–99 (cf. p. 33).

[128] *Uppaal*. URL : http://www.uppaal.org/ (cf. p. 104).

[129] A. Varshovi et B. Sadeghiyan. « Ontological Classification of Network Denial of Service Attacks : Basis for a Unified Detection Framework ». In : *Scientia Iranica* 17.2 (déc. 2010), p. 133–148 (cf. p. 36).

[130] Yaron Velner. « The Complexity of Mean-Payoff Automaton Expression ». In : *Proceedings of the 39th International Colloquium on Automata, Languages, and Programming (ICALP'12, Part II)*. Sous la dir. d'Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts et Roger Wattenhofer. T. 7392. Lecture Notes in Computer Science. Springer, juil. 2012, p. 390–402 (cf. p. 131).

[131] Yaron Velner, Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, Alexander Rabinovich et Jean-François Raskin. « The complexity of Multi-Mean-Payoff and Multi-Energy Games ». In : *CoRR* abs/1209.3234 (2012) (cf. p. 131, 142).

[132] Anthony D. Wood et John A. Stankovic. « A Taxonomy for Denial-of-Service Attacks in Wireless Sensor Networks ». In : *Handbook of Sensor Networks : Compact Wireless and Wired Sensing Systems*. Sous la dir. de Mohammad Ilyas et Imad Mahgoub. CRC Press, juil. 2004 (cf. p. 44).

[133] Kui Wu, Dennis Dreef, Bo Sun et Yang Xiao. « Secure Data Aggregation without Persistent Cryptographic Operations in Wireless Sensor Networks ». In : *Ad Hoc Networks* 5.1 (jan. 2007), p. 100–111 (cf. p. 30).

[134] Yang Xiao, Venkata Krishna Rayi, Bo Sun, Xiaojiang Du, Fei Hu et Michael Galloway. « A Survey of Key Management Schemes in Wireless Sensor Networks ». In : *Computer Communications* 30.11–12 (sept. 2007), p. 2314–2341 (cf. p. 33).

[135] Bashir Yahya et Jalel Ben-Othman. « Towards a Classification of Energy Aware MAC Protocols for Wireless Sensor Networks ». In : *Wireless Communications and Mobile Computing* 9.12 (déc. 2009), p. 1572–1607 (cf. p. 14, 16).

[136] Wei Ye, John Heidemann et Deborah Estrin. « An Energy-Efficient MAC Protocol for Wireless Sensor Networks ». In : *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'02)*. T. 3. New-York, NY, États-Unis, juin 2002, p. 1567–1576 (cf. p. 14).

[137] Ossama Younis et Sonia Fahmy. « HEED : A Hybrid, Energy-Efficient Distributed Clustering Approach for Ad-Hoc Sensor Networks ». In : *IEEE Transactions on Mobile Computing* 3.4 (oct. 2004), p. 366–379 (cf. p. 21, 103).

[138] Theodore Zahariadis, Panagiotis Trakadas, Helen C. Leligou, Sotiris Maniatis et Panagiotis Karkazis. « A Novel Trust-Aware Geographical Routing Scheme for Wireless Sensor Networks ». In : *Wireless Personal Communications* 69.2 (mar. 2013), p. 805–826 (cf. p. 41, 62).

[139] Madeleine El-Zaher, Jean-Michel Contet, Pablo Gruer, Franck Gechter et Abderrafiaa Koukam. « Compositional Verification for Reactive Multi-Agent Systems applied to Platoon Non Collision Verification ». In : *Studia Informatica Universalis* 10.3 (2012), p. 119–141 (cf. p. 22, 23).

[140] Yi-Hua Zhu, Wan-Deng Wu, Jian Pan et Yi-Ping Tang. « An Energy-Efficient Data Gathering Algorithm to Prolong Lifetime of Wireless Sensor Networks ». In : *Computer Communications* 33.5 (mar. 2010), p. 639–647 (cf. p. 17, 22).

[141] Sencun Zhu, Sanjeev Setia et Sushil Jajodia. « LEAP : Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks ». In : *Proceedings of the 10th ACM Conference on Computer and Communications Security (CSS'03).* Washington, DC, États-Unis, oct. 2003, p. 62–72 (cf. p. 31).

[142] Sencun Zhu, Sanjeev Setia et Sushil Jajodia. « LEAP+ : Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks ». In : *ACM Transactions on Sensor Networks* 2.4 (nov. 2006), p. 500–528 (cf. p. 31).

[143] ZigBee Alliance. *ZigBee Specification.* 2008 (cf. p. 32).

[144] Uri Zwick et Mike Paterson. « The Complexity of Mean Payoff Games on Graphs ». In : *Theoretical Computer Science* 158.1–2 (1996), p. 343–359 (cf. p. 142).

Index

Divers

A

B

C

O

P

R

S

T

U

V

W

Z

Liste des sigles

μTESLA *Micro version of the Timed Efficient Stream Loss-tolerant Authentication*

6LoWPAN *IPv6 over Low power Wireless Personal Area Networks*

AES *Advanced Encryption Standard*

ALH Automate Linéaire Hybride

AODV *Ad hoc On-demand Distance Vector routing*

B.A.T.M.A.N. *Better Approach To Mobile Adhoc Networking*

BS *Base Station*

CBC *Cipher Block Chaining mode*

CBC-MAC *Cipher Block Chaining Message Authentication Code*

CCM *Counter mode CBC-MAC*

CDMA *Code Division Multiple Access*

CH *Cluster Head*

CIFS *Short InterFrame Space*

CMTC Chaine de Markov à Temps Continu

CSMA *Carrier Sense Multiple Access*

CSMA/CA *Carrier Sense Multiple Access with Collision Avoidance*

CTL *Computation Tree Logic*

CTR *CounTeR mode*

CTS *Clear To Send*

DCF *Distributed Coordination Function*

DCS *Data Centric Sensor networks*

DDoS *Distributes Denial of Service*

DIFS *DCF InterFrame Space*

DLSR *DDoS-preventing OLSR*

DSDV *Destination-Sequenced Distance Vector routing*

DSR *Dynamic Source Routing*

DSSS *Direct-sequence Spread Spectrum*

EBS *Exclusion Basis System*

ERP-DCS *Efficient Rekeying Protocol for DCS sensor networks*

FDMA *Frequency Division Multiple Access*

FFUCA *Fast and Flexible Unsupervised Clustering Algorithm*

FHSS *Frequency-Hopping Spread Spectrum*

FTP *File Transfer Protocol*

HEED *Hybrid, Energy-Efficient Distributed clustering*

HTTP *HyperText Transfer Protocol*

IDS *Intrusion Detection System*

IEEE *Institute of Electrical and Electronics Engineers*

IETF *Internet Engineering Task Force*

IP *Internet Protocol*

KDM *Key Distribution Manager*

KMS *Key Management Submodule*

laser *Light Amplification by Stimulated Emission of Radiation*

LEACH *Low-Energy Adaptive Clustering Hierarchy*

LEAP *Localized Encryption and Authentication Protocol*

LLC *Logical Link Control*

LSAH Logique Stochastique avec Automates Hybrides

MAC *Media Access Control*

MAC *Message Authentication Code*

MANET *Mobile Ad hoc NETworks*

MLCG *Multiplicative Linear-Congruential Generators*

MPC *Multiple Parameter based Clustering*

MSQPS *Modified Secured Query Processing Scheme*

OCB *Offset CodeBook mode*

OLSR *Optimized Link State Routing protocol*

OSI *Open Systems Interconnection*

pDCS *privacy-enhanced DCS*

PSED Processus Stochastiques à Évènements Discrets

radar *RAdio Detection And Ranging*

RAM *Random Access Memory*

RC5 *Rivest Cipher version 5*

RFID *Radio Frequency IDentification*

RPSG Réseaux de Petri Stochastiques Généralisés

RPSGe Réseaux de Petri Stochastiques Généralisés étendus

RTS *Request To Send*

S-LEACH *Secure LEACH* (distinct de SecLEACH)

S-MAC *Sensor-MAC*

SDMA *Space Division Multiple Access*

SDMP *Securing Data based on Multi-Path routing*

SecCBSN *Secure Communications of Cluster-Based Sensor Network*

SecLEACH *Secure LEACH*

SNEP *Secure Network Encryption Protocol*

SPINS *Security Protocols for Sensor Networks*

SPRT *Sequential Probability Radio Test*

SSH *Secure SHell*

TCP *Transmission Control Protocol*

TDMA *Time Division Multiple Access*

TESLA *Timed Efficient Stream Loss-tolerant Authentication*

UDP *User Datagram Protocol*

VANET *Vehicular Ad hoc NETworks*

VSR *Virtual Struture Routing protocol*

WANET *Wireless Ad hoc NETwork*

WEP *Wired Equivalent Privacy*

WSN *Wireless Sensor Networks*

Crédits illustrations

Figures 4.15 à/*to* 4.23 :
Paolo Ballarini (© 2011)

Figures 7.4 à/*to* 7.7 :
Mathieu Sassolas (© 2014)

Ces illustrations sont soumises au droit d'auteur
de leur créateur respectif.

Those illustrations are subject to the copyright of their respective author.

Cet ouvrage a été entièrement conçu et réalisé à l'aide
des logiciels libres suivants :

biber	(bibliographie)
d3-cloud	(illustration : nuage de mots – 1.1)
Dia	(illustrations – 2.1, 2.4, 4.1, 4.2, 4.3, 4.4, 4.5, 5.1)
Git	(archivage des versions)
Gnuplot	(illustrations : graphiques – 4.7, 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, 5.4, 5.5, 5.6, 6.1, 6.2, 6.3, 6.4, 6.5, 6.6, 6.7, 6.9)
Graphviz	(illustration : machine à états – 5.2)
Inkscape	(illustrations – 4.6, 4.15, 4.16, 4.17, 4.18, 4.19, 4.20, 4.21, 4.22, 5.2, 5.3, 6.8, 6.10)
Vim	(édition)
Waf	(compilation)
XeTeX	(mise en page, illustrations – 2.2, 2.3, 3.1, 3.2, 4.14, 4.23, 7.1, 7.2, 7.3, 7.4, 7.5, 7.6, 7.7)

Les fichiers sources utilisés pour réaliser cet ouvrage sont disponibles à
l'adresse suivante :
https://github.com/Qeole/PhD

Composés d'appareils fortement limités en ressources (puissance de calcul, mémoire et énergie disponible) et qui communiquent par voie hertzienne, les réseaux de capteurs sans fil composent avec leurs faibles capacités pour déployer une architecture de communication de manière autonome, collecter des données sur leur environnement et les faire remonter jusqu'à l'utilisateur. Des « transports intelligents » à la surveillance du taux de pollution environnemental, en passant par la détection d'incendies ou encore l'« Internet des objets », ces réseaux sont aujourd'hui utilisés dans une multitude d'applications. Certaines d'entre elles, de nature médicale ou militaire par exemple, ont de fortes exigences en matière de sécurité.

Les travaux de cette thèse se concentrent sur la protection contre les attaques dites par « déni de service », qui visent à perturber le fonctionnement normal du réseau. Ils sont basés sur l'utilisation de capteurs de surveillance, qui sont périodiquement renouvelés pour répartir la consommation en énergie. De nouveaux mécanismes sont introduits pour établir un processus de sélection efficace de ces capteurs, en optimisant la simplicité de déploiement (sélection aléatoire), la répartition de la charge énergétique (sélection selon l'énergie résiduelle) ou encore la sécurité du réseau (élection démocratique basée sur un score de réputation). Sont également fournis différents outils pour modéliser les systèmes obtenus sous forme de chaines de Markov à temps continu, de réseaux de Petri stochastiques (réutilisables pour des opérations de model checking) ou encore de jeux quantitatifs.

Wireless sensor networks are made of small devices with low resources (low computing power, little memory and little energy available), communicating through electromagnetic transmissions. In spite of these limitations, sensors are able to self-deploy and to auto-organize into a network collecting, gathering and forwarding data about their environment to the user. Today those networks are used for many purposes : "intelligent transportation", monitoring pollution level in the environment, detecting fires, or the "Internet of things" are some example applications involving sensors. Some of them, such as applications from medical or military domains, have strong security requirements.

The work of this thesis focuses on protection against "denial of service" attacks which are meant to harm the good functioning of the network. It relies on the use of monitoring sensors : these sentinels are periodically renewed so as to better balance the energy consumption. New mechanisms are introduced so as to establish an efficient selection process for those sensors : the first one favors the ease of deployment (random selection), while the second one promotes load balancing (selection based on residual energy) and the last one is about better security (democratic election based on reputation scores). Furthermore, some tools are provided to model the system as continuous-time Markov chains, as stochastic Petri networks (which are reusable for model checking operations) or even as quantitative games.