Détection et diagnostic des fautes dans des systèmes à base de réseaux de capteurs sans fils

Dima Hamdan

▶ **To cite this version:**

Dima Hamdan. Détection et diagnostic des fautes dans des systèmes à base de réseaux de capteurs sans fils. Autre [cs.OH]. Université de Grenoble; Université Libanaise, 2013. Français. NNT : 2013GRENM007 . tel-00953241

HAL Id: tel-00953241

https://theses.hal.science/tel-00953241

Submitted on 28 Feb 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

préparée dans le cadre d'une cotutelle entre l'*Université de Grenoble* et l'Université Libanaise

Spécialité : INFORMATIQUE

Arrêté ministériel : le 6 janvier 2005 -7 août 2006

Présentée par

« Dima HAMDAN »

Thèse dirigée par **« Ioannis PARISSIS» et « Abbas HIJAZI »**
codirigée par **« Oum-El-Kheir AKTOUF» et « Bachar El HASSAN »**

préparée au sein des **Laboratoires**

Laboratoire de Conception et d'Intégration des Systèmes (LCIS)
« Laboratoire Systèmes électroniques, Télécommunications et Réseaux (LASTRE)

dans **les Écoles Doctorales**

Mathématiques, Sciences et Technologies de l'Information, Informatiques (MSTII)
Ecole Doctorale des Sciences et de Technologie (EDST)

Détection et diagnostic des fautes dans des systèmes à base de réseaux de capteurs sans fils

Thèse soutenue publiquement le **« 20 Février 2013 »**,
devant le jury composé de :

Mme Véronique Vèque
Professeur à l'université Paris Sud, Président
M. Yves Le traon
Professeur à l'université Luxembourg, Rapporteur
M. Hicham Snoussi
Professeur à l'université de technologie et de troyes, Rapporteur
M. Mourad Gueroui
Professeur à l'Université de Versailles, Membre
M. Ioannis Parissis
Professeur à Grenoble INP, Membre
Mme Oum-El-Kheir Aktouf
Maître de conférences à Grenoble INP, Membre
M. Abbas Hijazi
Professeur à l'université Libanaise, Membre
M. Bachar El Hassan
Maître de conférences à l'université Libanaise, Membre

REMERCIEMENTS

Mes premiers hommages vont à ceux qui m'ont encadrée au cours de ces trois années à LCIS (France) Mr. Ioannis Parissis et Mme. Oum- El- Kheir Aktouf de l'équipe CTSYS. Je les remercie pour leurs compétences scientifiques exceptionnelles mêlées toujours d'une gentillesse extrême, leur encadrement constructif, leur confiance, leur hospitalité, leur disponibilité, leur professionnalisme et leur caractère sérieux qui ne va pas au détriment de leur sens d'humour. Je les remercie pour leur assiduité dans le travail, leur engagement ainsi que leur contribution à avancer le travail de la recherche.

Et je tiens à remercier particulièrement M. Parissis qui a dirigé le travail jusqu'au bout. J'aimerais lui exprimer ma reconnaissance et ma plus profonde gratitude. Quant à Mme. Aktouf qui a toujours éveillé et aiguisé en moi l'amour du travail. Je la remercie pour les expressions étonnamment motivantes, pour la surveillance continue de mon travail (malgré la distance), les discussions enrichissantes, les idées judicieuses, sans lesquelles mon travail n'aurait pas eu la même valeur. Au terme des années de thèse, je me trouve fortunée (au dire des membres de LCIS et L'ESISAR) pour avoir travaillé avec une personne compétente, aimable et douée comme elle, *bref la meilleure directrice de thèse au monde.*

Mes sincères remerciements vont également à M. Abbas Hijazi, professeur à la faculté de Sciences (Hadath-Beyrouth). Je tiens à le remercier pour avoir accepté d'être le directeur de cette thèse. Je le remercie pour toutes ses contributions précieuses à réussir cette thèse : scientifique, administrative, financière (bourse AUF), les contacts avec des chercheurs travaillant dans le domaine (professeurs et doctorants), pour ses remarques constructives ainsi que pour ses qualités humaines (le support, les conseils, la gentillesse,..). J'apprécie fortement toutes ses initiatives et ses efforts pour mener au bout de ce travail.

Je remercie M. Bachar El Hassan (responsable de l'équipe TR du laboratoire LASTRE) et je serais heureuse si nous pouvions coopérer pour élaborer un travail de valeur.

Mes remerciements vont également aux directeurs du laboratoire LCIS (Grenoble INP, Valence, France) et du centre AZM pour la recherche en biotechnologie et ses applications (Université Libanaise, Tripoli, Liban) : Messieurs M. Eduardo Mendez et M. Mohammad Khalil. Je les remercie de leur accueil et de m'avoir permis de réaliser ce travail.

Je tiens également à remercier les membres de jury qui ont accepté d'évaluer ce travail :

M. Yves Le Traon, Professeur à l'Université du Luxembourg,
M. Hicham Snoussi, Professeur à l'Université de Technologie de Troyes,
Mme Véronique Vèque, Professeur à l'Université Paris Sud,
M. Mourad Gueroui, Professeur à l'Université de Versailles,

M. Ioannis Parissis, Professeur à Grenoble INP,
Mme Oum-El-Kheir Aktouf, Maître de Conférences à Grenoble INP,
M. Abbas Hijazi, Professeur à l'Université Libanaise,
M. Bachar El Hassan, Maître de Conférences à l'Université Libanaise.

Je voudrais remercier L'agence Universitaire de la francophonie (AUF) pour le financement de la thèse pendant les années académiques 2010-2011 et 2011-2012. Je remercie en particulier le directeur de l'agence à Tripoli M. Hassan Amoud et Rayan Nasser l'animatrice-documentaliste de l'agence pour l'intérêt dont ils ont fait montre à suivre les procédures administratives nécessaires pour que je puisse profiter de services fournis par l'AUF.

Je voudrais aussi remercier Mira Sarkis, l'étudiante qui a fait son stage de PFE sous l'égide de ma propre thèse. Je la remercie pour le travail sérieux et l'esprit du travail en groupe qu'elle a manifestés.

Je remercie Levent Gürgen, Yazid Benazzouz, Bassam Mosslem et Alia Ghaddar pour l'échange des idées faites autour du thème de ce travail.

Je voudrais remercier également M. Fadel Yakhni (directeur de la faculté de gestion à l'Université Libanaise) et Ezzeddine El- Eter (directeur du Lycée Saba Zreik) pour leur support ainsi que leur bon sens de compréhension au sujet de mon absentéisme (mes voyages en France).

Je voudrais aussi remercier tous mes enseignants pendant les parcours scolaire et universitaire, en particulier Mme Chehadeh et M. Chafic Mokbel qui m'ont apporté leur savoir et donc contribué de manière indirecte à la réalisation de cette thèse.

Mes remerciements vont également à tous mes amis et mes collègues au Liban (Cecilia Sassine, Najwa Kadamani, Bassima Mallat, Moulouk Mehrez, Bassima Rifaii...) ainsi que les amis que j'ai eu la chance de rencontrer en France : Darine Kaddour, Mazen Awad, Lina Hijazi, Noura Hatem, Khaoula Smaiili et Raji Nair. Que tous trouvent ici l'expression de ma gratitude pour les fructueux échanges quotidiens.

Je close enfin ces remerciements en dédiant cette thèse de doctorat à mes chers parents (mon père Sleimen et ma mère Awataf), à mes sœurs (Dalal, Hiyam, Mona, Hanadi et Suzi) et à mes frères (Ali, Mohammad, Ibrahim, Hassan, Oussama, Mazen et Majdi). Je leur suis infiniment reconnaissante pour leur soutien moral et leur encouragement sans lesquels je n'aurais pas pu tenir bon devant les difficultés que j'ai rencontrées.

RESUME

Les pannes sont la règle et non l'exception dans les réseaux de capteurs sans fil. Un nœud capteur est fragile et il peut échouer en raison de l'épuisement de la batterie ou de la destruction par un événement externe. En outre, le nœud peut capter et transmettre des valeurs incorrectes en raison de l'influence de l'environnement sur son fonctionnement. Les liens sont également vulnérables et leur panne peut provoquer un partitionnement du réseau et un changement dans la topologie du réseau, ce qui conduit à une perte ou à un retard des données. Dans le cas où les nœuds sont portés par des objets mobiles, ils peuvent être mis hors de portée de la communication. Les réseaux de capteurs sont également sujets à des attaques malveillantes, telles que le déni de service, l'injection de paquets défectueux, entraînant un comportement inattendu du système et *ainsi de suite*. En plus de ces défaillances prédéfinies (c'est-à-dire avec des types et symptômes connus), les réseaux de capteurs présentent aussi des défaillances silencieuses qui ne sont pas connues à l'avance, et qui sont très liées au système. En revanche, les applications de RCSF, en particulier les applications de sécurité critiques, telles que la détection d'incendie ou les systèmes d'alarme, nécessitent un fonctionnement continu et fiable du système. Cependant, la garantie d'un fonctionnement correct d'un système pendant l'exécution est une tâche difficile. Cela est dû aux nombreux types de pannes que l'on peut rencontrer dans un tel système vulnérable et non fiable. Une approche holistique de la gestion des fautes qui aborde tous les types de fautes n'existe pas. En effet, les travaux existants se focalisent sur certains états d'incohérence du système. La raison en est simple : la consommation d'énergie augmente en fonction du nombre d'éléments à surveiller, de la quantité d'informations à collecter et parfois à échanger. Dans cette thèse, nous proposons un «Framework » global pour la gestion des fautes dans un réseau de capteurs. Ce framework, appelé « IFTF », fournit une vision complète de l'état du système avec la possibilité de diagnostiquer des phénomènes anormaux. IFTF détecte les anomalies au niveau des données, diagnostique les défaillances de réseau, détecte les défaillances d'applications, et identifie les zones affectées du réseau. Ces objectifs sont atteints grâce à la combinaison efficace d'un service de diagnostic réseau (surveillance au niveau des composants), un service de test d'applications (surveillance au niveau du système) et un système de validation des données. Les deux premiers services résident sur chaque nœud du réseau et le système de validation des données réside sur chaque chef de groupe. Grâce à IFTF, les opérations de maintenance et de reconfiguration seront plus efficaces, menant à un système WSN (Wireless Sensor Network) plus fiable. Du point de vue conception, IFTF fournit de nombreux paramètres ajustables qui le rendent approprié aux divers types d'applications. Les résultats de simulation montrent que la solution présentée est efficace en termes de coût mémoire et d'énergie. En effet, le système de validation des données n'induit pas un surcoût de communication. De plus, le fonctionnement des deux services test et diagnostic augmente la consommation d'énergie de 4% en moyenne, par rapport au fonctionnement du service de diagnostic uniquement.

Mots clés : *réseau de capteurs ; tolérance aux fautes ; détection et diagnostic ; test ; consommation d'énergie.*

ABSTRACT

Sensor faults are the rule and not the exception in every Wireless Sensor Network (WSN) deployment. Sensor nodes are fragile, and they may fail due to depletion of batteries or destruction by an external event. In addition, nodes may capture and communicate incorrect readings because of environmental influence on their sensing components. Links are also failure-prone, causing network partitions and dynamic changes in network topology, leading to delays in data communications. Links may fail when permanently or temporarily blocked by an external or environmental condition. Packets may be corrupted due to the erroneous nature of communications. When nodes are embedded or carried by mobile objects, nodes can be taken out of the range of communications. WSNs are also prone to malicious attacks, such as denial of service, injection of faulty packets, leading to unexpected behavior of the system and so on. In addition to these predefined faults or failures (i.e., with known types and symptoms), many times the sensor networks exhibits silent failures that are unknown beforehand and highly system-related. *Applications over WSNs, in particular safety critical applications, such as fire detection or burglar alarm systems, require continuous and reliable operation of the system*. However, validating that a WSN system will function correctly at run time is a hard problem. This is due to the numerous faults that can be encountered in the resource constrained nature of sensor platforms together with the unreliability of the wireless links networks. A holistic fault management approach that addresses all fault issues does not exist. Existing work most likely misses some potential causes of system failures. The reason is simple : the more elements to monitor, the more information to be collected and sometimes to be exchanged, then the more the energy consumption becomes higher.

In this thesis, we propose an Integrated Fault Tolerance Framework (IFTF) that provides a complete picture of the system health with possibility to zoom in on the fault reasons of abnormal phenomena. IFTF detects data anomalies, diagnoses network failures, detects application level failures, identifies affected areas of the network and may determine the root causes of application malfunctioning. These goals are achieved efficiently through combining a network diagnosis service (component/element level monitoring) with an application testing service (system level monitoring) and a data validation system. The first two services reside on each node in the network and the data validation system resides on each cluster head. Thanks to IFTF, the maintenance and reconfiguration operations will be more efficient leading to a more dependable WSN. From the design view, IFTF offers to the application many tunable parameters that make it suitable for various application needs. Simulation results show that the presented solution is efficient both in terms of memory use and power consumption. Data validation system does not incur power consumption (communication overhead). Using testing service combined to diagnosis service incurs a 4 %, on average, increase in power consumption compared to using solely network diagnosis solutions**.**

Keywords: *wireless sensor networks; fault tolerance; detection and diagnosis; test, energy consumption.*

LISTE DES PUBLICATIONS

- D. Hamdan**,** O. Aktouf, I. Parissis, B. El Hassan, A. Hijazi. Integrated Fault tolerance for Wireless sensor networks. IEEE. 19th International Conference on Telecommunications, April 2012, Jounieh, Lebanon, pp. 1-6.
- D. Hamdan**,** O. Aktouf, I. Parissis, B. EL Hassan and A. Hijazi. Online Data Fault Detection for Wireless Sensor Networks - Case Study. IEEE. Third International Conference on Wireless Communication in Unusual and Confined Areas, August 2012, Clermont-Ferrand, France.
- D. Hamdan**,** O. Aktouf, I. Parissis, A. Hijazi, M. Sarkis, B. EL Hassan. Smart diagnosis service for wireless sensor networks. IEEE. 6th International Conference on Next Generation Mobile Applications, Services and Technologies, September 2012, Paris, France, pp. 211 – 216.
- D. Hamdan**,** O. Aktouf, I. Parissis, B. EL Hassan, A. Hijazi, B. Moslem. Self monitoring Adaptative and Resource Efficient service for Improving QOS in Wireless Sensor Networks. IEEE. International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, October 2012, Sanya, China.
- D. Hamdan, O. Aktouf, I. Parissis, A. Hijazi, B. EL Hassan. Runtime quality services assurance for Wireless sensor and actuator networks. 18th International Science Meeting, March 2012, Zouk Mikael, Lebanon.
- M. Sarkis, D. Hamdan, B. EL Hassan, O. Aktouf et I. Parissis. Online data fault detection in wireless sensor networks IEEE. Second International Conference on Advances in Computational Tools for Engineering Applications, December 2012, Zouk Mosbeh, Lebanon.
- M. Sarkis, D. Hamdan, O. Aktouf et I. Parissis. Perceptron: Détection Des Fautes De Données Dans Les RCSF. 9ème Conférence Internationale Jeunes Chercheurs, Octobre 2012, Lille, France.

LISTE DES FIGURES

LISTE DES TABLEAUX

TABLE DES MATIERES

Chapitre 1.

INTRODUCTION

Les systèmes à base de réseaux de capteurs sans fil (RCSF) ont été classés parmi les 21 technologies les plus importantes du 21éme siècle[1]. Cette nouvelle technologie suscite un intérêt croissant étant donné la diversité de ces applications. Cependant, la réalisation d'une application à base de ces systèmes pose des nombreux défis liés aux enjeux de la sûreté de fonctionnement de ces systèmes. Dans ce chapitre, nous présentons ces principaux défis ainsi le cadre de notre travail de thèse.

SOMMAIRE

1.1. Contexte de la thèse

Le sujet de cette thèse s'articule autour de la sûreté de fonctionnement des systèmes à base de réseaux de capteurs sans fils (RCSF). En effet, le sujet prend son importance du fait du fort potentiel des applications autour des RCSF [2][3][4][5] (surveillance de l'environnement, détection de feux dans des grandes zones forestières, suivi de la croissance des plantes, télésurveillance de champs de bataille, surveillance à distance des patients, détection des inondations, suivi les structures d'avions, suivi les altérations de structure d'un bâtiment, détection des fuites de produits toxiques, *etc*.) et de la nature vulnérable de ce type du réseau. Cette importance s'accroît considérablement lorsqu'il s'agit d'applications critiques qui ont un impact direct sur l'homme ou sur l'environnement (surveillance des malades, détection des incendies ou pollution, *etc*.). Le résultat attendu est une approche de sûreté de fonctionnement qui rend les systèmes à base de RCSF plus robustes et tolérants aux fautes en tenant compte des contraintes de limitation de ressources (mémoire et énergie). Dans cette perspective, notre travail de recherche s'articulera autour de deux axes : la *tolérance aux fautes* et la *maîtrise de la consommation énergétique*.

La *tolérance aux fautes* a connu une importance considérable parmi les différents domaines de recherche sur les réseaux de capteurs sans fil ; en raison des contraintes d'énergie, d'environnement et de déploiement. Ce dernier étant d'un coût très élevé, il représente un handicap pour la réorganisation du réseau en cas de panne d'un ou plusieurs de ses capteurs, d'où l'importance d'introduire un mécanisme de tolérance aux fautes afin de garantir le bon fonctionnement du réseau notamment après la défaillance de certains de ses composants.

Par ailleurs, la *maîtrise de la consommation énergétique* représente un facteur déterminant dans la vie d'un réseau de capteurs parce que les nœuds disposent en général de ressources limitées en énergie et il est impossible de remplacer ou bien de recharger un nœud en fin de vie. Ceci rend l'optimisation de l'énergie beaucoup plus compliquée parce que il ne s'agit pas seulement de la réduction de la consommation mais aussi du prolongement de la vie d'un réseau.

1.2. Problématiques

Les problématiques qui nous intéressent plus particulièrement dans le contexte donné ci-dessus s'articulent autour des trois points suivants.

1.2.1. Couverture des fautes

La limitation d'énergie dans les capteurs sans fil, et les environnements hostiles dans lesquels ils pourraient être déployés, sont des facteurs qui rendent ce type de réseaux très vulnérables. Ainsi, la perte de connexions sans fils peut être due à une extinction d'un capteur suite à un épuisement de sa batterie, ou tout simplement à une destruction physique accidentelle ou intentionnelle par un ennemi [6][6][7]. Par ailleurs, l'absence de sécurité physique pour ce type de capteurs, et la nature vulnérable des communications

radios sont des caractéristiques qui augmentent les risques de pannes sur ce type de réseau [8], *etc*.

Etant donné le risque élevé et la grande diversité des types des pannes, le RCSF devrait être articulé autour de deux tâches : une tâche consiste à fournir les fonctionnalités de l'application (récolter et transmettre des données, envoyer des alarmes pour permettre la prévention des situations critiques) et une autre tâche qui consiste à surveiller le système, en particulier l'état des capteurs et des connexions. Toutefois, la disponibilité des ressources (mémoire et énergie) nécessaires pour réaliser la deuxième tâche est limitée. Ainsi, les coûts vont conditionner la stratégie de sûreté de fonctionnement du système. Une question importante se pose ici : Comment peut-on assurer une couverture élargie des types de pannes en utilisant le moins possible les ressources limitées des capteurs ?

En effet, le fonctionnement correct de systèmes à base de RCSF est lié principalement à trois aspects fondamentaux : la fiabilité des mesures collectées de l'environnement, la fiabilité des différents éléments qui entrent dans la composition d'un tel système (liens de communication et nœuds) et la fiabilité de l'application finale. Une solution *globale* devrait traiter ces aspects ensemble tout en maîtrisant la consommation d'énergie.

1.2.2. Hétérogénéité

Les réseaux de capteurs sont collaboratifs et très orientés vers un domaine d'application précis. Chaque type d'applications nécessite des mécanismes de sûreté de fonctionnement adéquats avec son environnement. Pour cela on trouve des mécanismes plus efficaces pour certaines applications que pour d'autres. En outre, de nombreux capteurs de différents types, de différents fournisseurs sont déjà déployés dans diverses applications. Actuellement, le traitement des données de capteurs est en général spécifique à l'application. Chaque application a ses propres capteurs avec des méthodes de traitement spécifiques.

Avec la grande diversité des capteurs et des applications liés à ce type de réseaux, il est très difficile de trouver des solutions de sûreté de fonctionnement génériques dont les spécifications sont totalement indépendantes des applications et des capteurs. En effet, une solution *générale* peut être exploitée dans de nombreux contextes.

1.2.3. Passage à l'échelle

Le nombre de capteurs déployés peut être de l'ordre du millier ou de la centaine de milliers, et vraisemblablement plus dans un futur proche. De plus, les réseaux de capteurs sont constitués sans aucune topologie prédéterminée. Ainsi, les mécanismes de sûreté de fonctionnement dédiés aux réseaux de capteurs doivent être capables de fonctionner efficacement avec une grande quantité de capteurs. Ces mécanismes doivent être capables de traiter un grand nombre d'événements sans être saturés.

Pour effectuer une tâche de surveillance, de diagnostic ou de débogage, les nœuds génèrent souvent des paquets supplémentaires à leurs propres paquets de données. Ces paquets supplémentaires sont généralement appelés paquets de contrôle. Si dans un réseau classique, le nombre de paquets de contrôle est presque négligeable, il ne l'est pas dans un réseau de capteurs. En effet la surcharge du réseau augmente avec le nombre de nœuds et donc davantage de paquets de contrôle sont nécessaires pour

surveiller le système. Comme la bande passante du réseau est limitée, la surcharge du réseau peut réduire la bande passante utilisée pour la transmission des données. Par conséquent, l'accroissement du nombre de nœuds du réseau réduit la bande passante disponible pour la transmission des données.
Le déploiement des réseaux de capteurs soulève alors une question intéressante quant à leur fiabilité. Comme un réseau de capteurs est conçu pour s'adapter à un environnement en évolution, il émettra plus de paquets de contrôle face à un changement environnemental. À un certain point, le réseau ne pourra plus maintenir la surcharge, et la transmission de données deviendra de moins en moins fiable. En effet, le passage à l'échelle et la fiabilité sont étroitement liés et peuvent se perturber mutuellement.

1.3. Objectifs et démarches de l'étude

L'objectif de cette thèse est de proposer une approche de la gestion des fautes globale, générale, et à grande échelle des systèmes à base de réseaux de capteurs sans fils. Pour cela, notre démarche a été la suivante :

Lors d'une première étape de notre travail, nous avons débuté par une recherche bibliographique sur les divers travaux de tolérance aux pannes qui ont été effectués, et qui se font à l'heure actuelle dans les systèmes de capteurs. L'objectif de cette première étape est d'appréhender les différents types de fautes qui affectent les éléments du RCSF, les principales approches existantes tant au niveau de la tolérance aux fautes que de la gestion de l'énergie. Ensuite, nous avons focalisé notre travail sur trois classes de fautes (données, réseau et application) qui ont un impact sur la performance du système à base de RCSF. Nous avons proposé des solutions pour traiter chaque classe de fautes retenue. Nous avons considéré la contrainte forte de limitation de ressources (mémoire et énergie) qui conditionne la validité de chaque solution. Finalement, nous avons défini une architecture globale fondée sur l'intégration de l'ensemble des solutions proposées et validées.

1.4. Contributions

Après une étude exhaustive de l'état de l'art dans chaque étape de notre travail, nous avons conclu qu'une approche globale de la tolérance aux fautes n'existe pas. En effet, la démarche de conception d'une telle approche consiste à imaginer « toutes » les fautes possibles et à prévoir une réponse adaptée permettant la préservation des fonctions essentielles du système. Le problème étant bien entendu qu'il n'est pas possible d'imaginer et de prévoir un traitement pour « toutes » les fautes. Celles-ci sont en effet en nombre potentiellement infini (si l'on considère les fautes survenant dans les composants matériels) et de toutes façons très grand (tout « bug » est une faute !). De plus, il convient également de prévoir l'occurrence simultanée des fautes élémentaires identifiées.

De ce fait, la principale contribution de cette thèse est la proposition d'une approche globale et générale de la gestion des fautes adaptée au RCSF. Cette approche traite les fautes à plusieurs niveaux (données, nœuds/réseau, et services d'application) à travers

une architecture unique qui permet l'orchestration des services complémentaires et orthogonaux. Les contraintes principales de notre travail ont été :

- La maîtrise de la consommation des ressources en présence de plusieurs types de services de détection des fautes fonctionnant en ligne ;
- L'assurance de la couverture élargie et potentielle des fautes en temps réel ;
- La conception d'un mécanisme qui permet l'intégration de ces différents services dans un seul Framework ;
- La réalisation de la généralité de l'approche vis-à-vis du type d'application ainsi que du type de capteur utilisé.

Pour atteindre l'objectif visé, nous avons étudié la pertinence des trois approches de détection des fautes :

- Un système de détection et de classification des fautes au niveau des données capteurs.
- Un service de diagnostic des fautes au niveau nœud et réseau.
- Un service de test au niveau de l'application.

1.5. Organisation du document

Ce document de thèse est organisé comme suit :

- Après ce chapitre qui est une introduction générale de nos travaux, des problématiques et des objectifs de cette thèse, le chapitre 2 a comme objectif la présentation d'une étude de l'état de l'art couvrant la sûreté de fonctionnement dans les systèmes à base de RCSF. Dans un premier temps, nous décrivons l'architecture matérielle et logicielle de ces systèmes. Dans un second temps, nous présentons des modèles de la tolérance aux fautes fondées sur la redondance. Nous détaillons deux catégories d'approches d'inspection des fautes dans le RCSF. Une catégorie adresse la détection et le diagnostic des fautes aux niveaux suivants : données, nœud et réseau. L'autre classe traite le test et de débogage au niveau application. Nous présentons le potentiel d'inspection de chaque classe par une étude comparative des approches de deux classes. Finalement, nous présentons des approches d'évitement des fautes par des techniques de conservation d'énergie.

- Le chapitre 3 décrit le premier pas vers la définition d'une approche globale de la gestion des fautes. Dans ce chapitre, nous présentons tout d'abord les spécificités des données capteurs. En se basant sur ces spécificités, nous proposons trois méthodes de détection des fautes. Nous décrivons notre méthodologie de travail ainsi que la base des données utilisée pour le travail expérimental. Nous discutons les résultats de chaque méthode. Finalement, ce chapitre est terminé par une proposition d'un système hybride dont l'avantage principal est de réduire les taux de faux négatifs et/ou positifs de détection des fautes.

- Le chapitre 4 présente le service de diagnostic adopté en vue de concevoir une approche globale de la tolérance aux fautes. Dans ce chapitre, nous décrivons les caractéristiques et les fonctionnalités de ce service, son architecture, ainsi que la procédure de détection et de diagnostic des fautes dans ce service. Nous présentons le travail expérimental sur un simulateur qui s'appelle TOSSIM conçu pour les RCSF. Tout d'abord, nous détaillons l'environnement de simulation, en particulier le modèle de consommation d'énergie. Finalement, nous présentons les résultats expérimentaux qui valident l'efficacité du service.

- Le chapitre 5 présente notre approche de tolérance des fautes IFTF. Ce chapitre présente la principale contribution de cette thèse. Tout d'abord, nous présentons le potentiel de l'approche à partir d'un scenario d'application de RCSF. Ensuite, nous présentons l'architecture logicielle de l'approche. Dans ce chapitre, nous introduisons et présentons un service de test d'applications et un composant de gestionnaire des fautes qui font partie de l'architecture IFTF. Finalement, nous validons notre approche par une étude sur le coût mémoire et énergétique.

Le dernier chapitre est consacré aux conclusions générales obtenues de l'ensemble de ces travaux et aux perspectives de cette thèse.

Chapitre 2.

CADRE DES TRAVAUX ET ETAT DE L'ART

Depuis l'émergence des applications orientées capteurs, la sûreté de fonctionnement dans les systèmes à base de RCSF est devenue un domaine de recherche très important. D'une part, le risque de fautes élevé en fait des systèmes non fiables. D'autre part, la mise en œuvre des mécanismes de sûreté de fonctionnement posent de nombreux enjeux liés aux particularités spécifiques de ces systèmes. Ce chapitre présente les réseaux de capteurs ainsi que les travaux de recherche en sûreté de fonctionnement dans les RSCF.

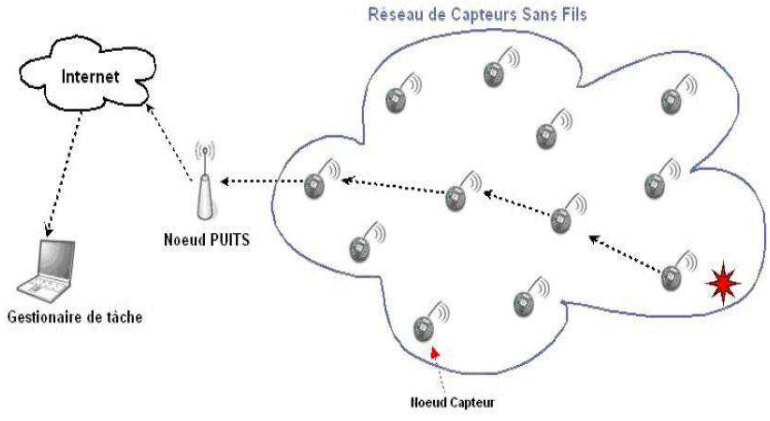
SOMMAIRE

2.1.Réseaux de capteurs sans fils

2.1.1. Architecture du réseau

Un réseau de capteurs sans fils[1] (Wireless Sensor Network ; WSN) est un type spécial de réseaux ad-hoc défini par un ensemble coopérant de nœuds capteurs dispersés dans une zone géographique appelée zone de captage afin de surveiller un phénomène et récolter les données d'une manière autonome. Ils sont construits autour des deux entités suivantes (Figure 2-1) :

- *Le capteur « sensor »* qui se charge de capter des données (de type son, vibration, optique, thermique, *etc*.), de traiter les informations à l'aide des valeurs collectées et d'envoyer des résultats de calcul à travers le réseau. Divers types de capteurs de température, de pression, de localisation, chimiques, audiovisuels, *etc*. sont déjà utilisés dans divers domaines d'applications : applications militaires, la surveillance de personnes malades à domicile, la sécurité, l'alerte sur des détections d'anomalies comme des incendies ou la pollution, le contrôle environnemental, *etc*. Les capteurs sont généralement statiques. Cependant certaines architectures réseau introduisent le concept de mobilité [10] dont l'intérêt est multiple dans la mesure où les capteurs mobiles peuvent notamment permettre de prolonger la vie du réseau, d'étendre la couverture d'un réseau, d'améliorer ses performances de routage ou sa connectivité globale.
- *Le puits « sink »* est le nœud final du réseau. C'est à lui qu'est envoyé l'ensemble des valeurs mesurées par le réseau. Il peut transférer les données collectées via internet ou par satellite à un ordinateur central « gestionnaire de tâches » ou *« station de base »* pour leur traitement. Il peut arriver qu'il y ait plusieurs puits sur un même réseau de capteurs [11]. En effet, une défaillance du puits provoque la mise hors service du réseau complet.
- *L'agrégateur (aggregator)* **:** il est facultatif dans les réseaux de capteurs. Il est en charge d'agréger les messages qu'il reçoit de plusieurs capteurs puis de les envoyer en un seul message au puits (sink). Cette opération a pour principal but de limiter le



2: Architecture des réseaux de capteurs sans fils [1]

trafic sur le réseau et donc de prolonger la durée de vie globale du réseau de capteurs. L'agrégateur correspond généralement au « clusterhead » dans un réseau d'architecture hiérarchique. L'utilisation de « clusters » offre de nombreux avantages à tous les niveaux, notamment pour le routage [12].

L'environnement du RCSF peut être *extérieur* par exemple pour des applications urbaines, *intérieur* par exemple pour les applications domotiques comme les maisons intelligentes, ou *hybride* intégrant les deux.

2.1.2. Plateformes

Suivant le type d'application, il y a plusieurs catégories de capteurs différents qui varient en fonction de la taille, de la capacité de calcul, de la taille de la mémoire et de la bande passante. Suivant les caractéristiques des capteurs, la taille du réseau, des protocoles de communications. Un RCSF est structuré suivant quatre types principaux de plateformes [13] :

- *Plateforme de capteurs miniaturisés* : plateforme dédiée aux capteurs de taille très réduite (quelques mm^3) et de faible bande passante (<50Kbps). Un exemple de ce genre de plateforme est Spec [14]. Avec une taille très petite (2mmx2.5mm), Spec est un des plus petits capteurs au monde.
- *Plateforme de capteurs généraux* : plateforme développée pour capter et router des informations du monde ambiant. Quelques plateformes de cette famille ont été développéss et la plus récente est basée sur MicaZ [15], un capteur de taille ~$10cm^3$ avec les protocoles de communication IEEE 802.15.4 [16]. Aujourd'hui, MicaZ devient la référence dans les travaux de recherche dans le domaine des réseaux de capteurs. Dans notre travail, nous nous intéressons à la plateforme de capteurs généraux.
- *Plateforme de capteurs à haute bande passante* : ces plateformes ont pour but de transporter de gros volumes de données captées (la vidéo, le son, les vibrations). Un exemple de cette famille est Imote [17] dont la communication se base sur la norme Bluetooth 1.1 [18].
- *Plateforme de passerelles* : ces dispositifs servent à transporter les informations envoyées par le réseau de capteurs vers un réseau traditionnel (Ethernet, 802.11) dont Stargate [15] est un exemple typique.

2.1.3. Systèmes d'exploitation

La gestion de l'énergie est un facteur important dans la conception d'un SE pour capteurs. Selon Wei Dong [19], les techniques actuelles pour économiser l'énergie sur des nœuds peuvent être différenciées entre la gestion énergétique des microcontrôleurs (calcul d'état de veille minimale [20]) et la gestion énergétique des périphériques (arrêt individuel d'un périphérique quand il n'est pas utilisé). De nombreux systèmes d'exploitation spécialisés existent, parmi lesquels :

- *TinyOS* [21] : est basé sur une architecture à composants, et son exécution est dirigée par des événements. Il est léger (moins de 400 octets en RAM [22])

flexible grâce à l'approche à composants et adapté pour l'exécution à basse consommation d'énergie. La librairie TinyOS comprend les protocoles réseaux, les services de distribution, les drivers pour capteurs et les outils d'acquisition de données. TinyOS est en grande partie écrit en C mais on peut très facilement créer des applications personnalisées en langages C, NesC, et Java.

- *Mantis[23]* : est un système multithreads développé en C, langage choisi pour son efficacité et sa portabilité. Son empreinte mémoire est faible : 500 octets en RAM et 14 ko en flash. L'économie d'énergie est réalisée par Mantis par une fonction de veille qui désactive le capteur lorsque tous les threads actifs sont terminés.
- *LiteOS* [24] : LiteOS fournit un environnement comparable à Unix adapté aux capteurs en réseau. Un langage de programmation orienté objet C++ est disponible pour développer des programmes et permettre leur déploiement sur les capteurs. LiteOS possède une faible empreinte mémoire et peut fonctionner avec 4 ko de RAM et 128 Ko de mémoire flash.
- *Contiki [25] :* le code est écrit en C. Le système complet est supposé pouvoir tourner sans problème avec 2 ko de RAM et 40 ko de mémoire flash.

Les résultats expérimentaux montrent que le système TinyOS est le système le plus économe en énergie en comparaison avec les systèmes mentionnés ci-dessus [26].

2.1.4. Applications

Les réseaux de capteurs sans fil font partie des thèmes de recherche très actifs actuellement car cette technique peut être appliquée dans de nombreux domaines. Quelques exemples d'applications sont listés ici.

- *Surveillance de catastrophes naturelles* : Des capteurs peuvent signaler des événements tels que feux de forêts, tempêtes, inondations, *etc*. Ceci permet une intervention beaucoup plus rapide et efficace des secours [27][28][29].
- *Surveillance médicale :* En implantant sous la peau de mini capteurs vidéo, on peut recevoir des images en temps réel d'une partie du corps sans aucune chirurgie pendant environ 24h. On peut ainsi surveiller la progression d'une maladie ou la reconstruction d'un muscle [30][31].
- *Applications industrielles* : les industriels s'intéressent au potentiel des capteurs pour diminuer les coûts du contrôle et de la maintenance des produits, de la gestion de l'inventaire, de la télésurveillance après vente, *etc*. [32].
- *Applications militaires :* comme pour plusieurs domaines, les applications militaires ont été aussi les locomotives de la recherche pour les réseaux de capteurs. Détection des troupes ou des véhicules d'ennemis était le sujet d'une des premières applications des réseaux de capteurs [33].

Selon les interactions entre le réseau de capteurs et le puits, on a trois modèles d'application principaux [34] :

- *Modèle de mesure périodique « Time-driven » [35]*. Tous les capteurs envoient périodiquement leurs mesures à la station de base. Le type d'application visé concerne les applications de type "surveillance" où le but principal est d'avoir une information régulière de la zone surveillée.
- *Modèle de détection d'événements « event-driven »[36]*. Les capteurs dans ce modèle envoient les mesures seulement lorsqu'il y a un événement qui se produit. Ce type de modèle est recommandé pour les applications de surveillance d'événements critiques où le but principal est l'obtention d'une information sur l'événement le plus rapidement possible.
- *Modèle de transmission suite à des requêtes « query-based »[37]*. Les capteurs mesurent des phénomènes et stockent ces mesures dans leur mémoire flash. Ils envoient ces mesures seulement lorsqu'ils reçoivent des requêtes de la station de base. Ce modèle peut également s'apparenter aux applications de type surveillance mais les capteurs utilisent une mémoire flash importante afin de stocker les mesures localement.

2.1.5. Caractéristiques

Même si les réseaux de capteurs partagent certaines des caractéristiques des réseaux *ad hoc* sans fils, ils ont des caractéristiques spécifiques, notamment au niveau des ressources. Parmi ces caractéristiques, nous citons :

- *Les ressources limitées* **:** la taille des capteurs peut varier de quelques centimètres jusqu'à quelques millimètres cubes. Naturellement, ceci limite les capacités des capteurs au niveau du calcul, du stockage, et de la communication, malgré l'avancement remarquable dans la technologie de miniaturisation. En conséquence, il est nécessaire que les algorithmes et les protocoles utilisent ces ressources très efficacement. En particulier, l'énergie est la ressource la plus précieuse des capteurs. Typiquement, un capteur est équipé d'une batterie (e.g. de type AAA) qui peut durer plusieurs heures en mode actif et plusieurs mois en mode veille. La durée de vie du système entier dépend du niveau d'énergie des capteurs individuellement. En effet, comme les capteurs participent au routage, à l'agrégation des résultats, ou à l'auto-configuration du réseau, l'indisponibilité de certains nœuds peut provoquer le dysfonctionnement de tout le système.
- *Environnement dynamique* : Tout au long de la durée de vie d'un réseau de capteurs sans fil, des changements environnementaux peuvent se produire. Les nœuds peuvent être déplacés en raison d'une forte vitesse du vent. De grands objets physiques peuvent être placés de sorte qu'ils interfèrent sur la communication entre les nœuds. Face à ces contraintes environnementales, la solution de la tolérance aux fautes doit être capable de réagir automatiquement afin de supporter ces événements.
- *L'échelle* **:** Le nombre de nœuds déployés pour un projet peut atteindre le million. Un nombre aussi important de nœuds engendre beaucoup de transmissions inter nodales et nécessite que le puits "sink" soit équipé de suffisamment de mémoire pour stocker les informations reçues. Les protocoles du RCSF et les mécanismes de tolérance aux fautes doivent donc prendre en compte le facteur "grande échelle" des réseaux de capteurs.

– *Le manque de fiabilité* : certains capteurs peuvent générer des erreurs ou ne plus fonctionner à cause d'un manque d'énergie, d'un problème physique ou d'une interférence. La transmission radio peut être momentanément perturbée par des champs électriques, *etc*. Par ailleurs, l'absence de sécurité physique pour ce type de capteurs et la nature vulnérable des communications radios sont des caractéristiques qui augmentent les risques de pannes sur ce type de réseaux. Un autre problème rencontré en traitant les données de capteurs est leur nature parfois incomplète et bruitée, et parfois redondante. En effet, les liens de communication qui sont non fiables, et des obstacles dans l'environnement causent des pertes, des collisions et des interférences sur les paquets envoyés par les capteurs. Ces incidents réduisent gravement la qualité des données.

En tenant compte ces caractéristiques, il sera nécessaire d'introduire des mécanismes de sûreté de fonctionnement qui rendent les systèmes à base de RCSF plus robustes et tolérants aux fautes.

2.2.Sûreté de fonctionnement de RCSF

Un système correct est un système qui réalise ce pour quoi il a été conçu. La maîtrise complète de la sûreté de fonctionnement d'un système consiste à connaître, évaluer, prévoir, mesurer et maîtriser les fautes de manière à garantir la fiabilité, la sécurité, la disponibilité et la maintenabilité du système. D'une part, il faut traiter le plus tôt possible les fautes avant qu'elles aient pu produire une erreur. En plus, il faut assurer la fourniture du service malgré l'occurrence de fautes, c'est ce qu'on appelle la *tolérance aux fautes*. Dans un RCSF, certains nœuds capteurs peuvent être bloqués ou tomber en panne à cause d'un manque d'énergie, d'un dégât matériel ou d'une interférence environnementale. La panne d'un nœud capteur ou d'un lien ne doit pas affecter le fonctionnement global du réseau.
La procédure de tolérance aux pannes dépend généralement de l'architecture et des fonctionnalités du système. Cependant, certaines étapes générales sont exécutées dans la plupart des systèmes, telles que : la détection de fautes, phase durant laquelle on reconnaît qu'un événement inattendu s'est produit, et le recouvrement de fautes phase dans laquelle on effectue des opérations d'élimination des effets de fautes. Le recouvrement des fautes dépend des types de fautes. Parmi les techniques utilisées, nous pouvons citer la redondance (faute permanente), la réinitialisation (faute transitoire), la reconfiguration des paramètres (faute opérationnelle), la reprogrammation (faute de conception logicielle), *etc*. Dans la suite, nous présentons des exemples de tolérance aux fautes fondée sur la redondance, puis nous détaillerons les stratégies de diagnostic et les approches de test d'applications dans les RCSF.

2.2.1. Quelques exemples d'approches de redondance

L'approche la plus classique pour la tolérance aux fautes dans les réseaux de capteurs est la redondance [38]. Les réseaux de capteurs sont généralement denses et redondants. En effet, suivant l'application, on déploiera plus ou moins de capteurs dans un souci d'allongement de la durée de vie de l'application et d'amélioration de la tolérance aux pannes. Cette technique est très efficace mais également très

coûteuse, car il faut prévoir plusieurs capteurs pour chacun des éléments à surveiller. Dans ce qui suit, nous décrivons quelques exemples de travaux utilisant les technqiues de redondance dans les RCSF.

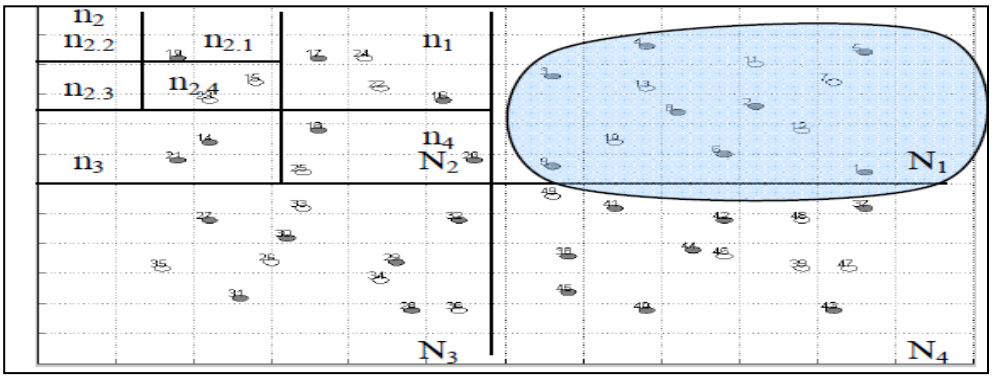


Figure 2-1 : L'architecture du réseau avec partitionnement de la zone de captage [39]

2.2.1.1. Redondance matérielle des nœuds

Certaines approches de redondance introduisent *un mécanisme d'endormissement* des nœuds [39][40] afin de prolonger la durée de vie du réseau. Ce mécanisme nécessite néanmoins la mise en place de méthodes d'ordonnancement des activités. Une de ces approches est présentée dans [39]. Elle adopte une technique de détection d'un capteur défectueux et sa substitution par un nœud dans l'état endormi. La technique de détection consiste à diviser successivement la zone de captage en sous-zones afin de trouver le capteur défectueux (Figure 2-1). Au départ, on divise le réseau en quatre zones disjointes dont chacune a un nœud maître (celui qui a l'ID le plus élevé ou éventuellement un niveau d'énergie le plus élevé). Celui-ci se charge de calculer périodiquement le débit de sa propre zone. S'il détecte que le débit réel est inférieur à un seuil prédéterminé de la zone, il divise la zone en quatre quadrants. Le processus de division et de test de débit se répète dans chaque quadrant jusqu'à atteindre un quadrant qui ne contient qu'un seul nœud, considéré alors comme étant un nœud défaillant puisqu'il dégrade le débit de la zone. La technique de substitution choisit dans le voisinage du nœud suspect un nœud endormi en bon état (qui a un débit plus élevé que le nœud suspect à l'éveil). La simulation montre l'efficacité de l'algorithme à identifier plusieurs capteurs défectueux dans une même zone et à détecter les pannes dans plusieurs zones simultanément. Cet algorithme a une complexité de l'ordre de O(n) où n est le nombre total des capteurs du réseau.

2.2.1.2. Redondance Matérielle – Sauvegarde des données

Une solution de la tolérante aux pannes des nœuds capteurs et *puits* est présentée dans [40]. L'objectif de l'algorithme est d'assurer une meilleure qualité

des données (QdD) avec une surcharge minimale de l'énergie. La qualité de données est mesurée par le rapport suivant :

$$QdD = \frac{nombre\ des\ paquets\ reçus\ par\ l'utilisateur\ \text{à}\ chaque\ période\ T}{nombre\ des\ paquets\ envoyés\ par\ les\ noeuds\ pendant\ T} \qquad (2.1)$$

L'algorithme suppose que les nœuds soient homogènes en termes de ressources donc n'importe quel capteur peut prendre le rôle du puits. De plus, il introduit un *mécanisme d'endormissement* pour prolonger la vie du réseau. Pour cela, dans chaque zone limitée de captage (E_i), il suffit d'avoir un capteur source au minimum qui prend en charge la transmission des données au puits. L'algorithme utilise la technique du « data checkpointing » et la sauvegarde incrémentale de la mémoire du puits dans la mémoire d'un autre capteur. Celui-ci remplace le puits dans l'agrégation et la transmission des données à l'utilisateur final lorsque le puits tombe en panne.

Pour cela, chaque nœud puits S_m envoie à chaque période (T_m) à un nœud prédécesseur S_{m-1} les données qu'il maintient d'une façon incrémentale. Si le niveau de la batterie du puits arrive à une valeur seuil prédéterminée, il envoie un message de contrôle au nœud S_{m-1} indiquant qu'il est incapable de remplir les fonctions de puits. Si le niveau de batterie du nœud S_{m-1} est suffisant (supérieur à la valeur seuil), il prend le rôle du puits. Sinon, il envoie le message de contrôle au nœud S_{m-2} et ainsi de suite jusqu'à la fin du chemin de « checkpoint » (Figure 2-2). La longueur du chemin est un paramètre de l'algorithme et affecte considérablement la consommation d'énergie. Si la longueur du chemin est égale à trois, cela signifie que la sauvegarde se fait à trois niveaux (S4 -> S3, S3->S2, S2->S1).

Une deuxième indication de la panne induite par un capteur S_{i-1} est réalisée en estimant le délai nécessaire pour recevoir les données du capteur suivant S_i. Le mécanisme de tolérance aux pannes proposé a l'avantage de réduire la perte des paquets collectés par le puits et de réduire le temps pendant lequel le réseau des capteurs reste sans puits.

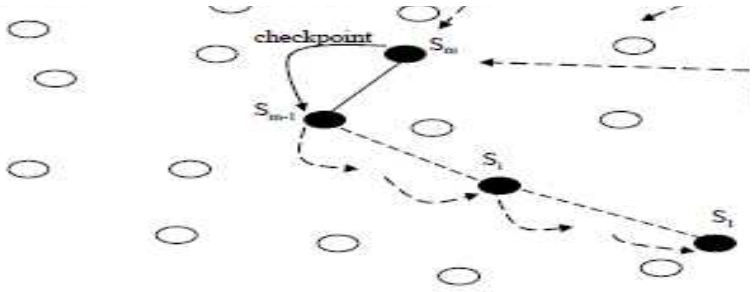


Figure 2-2: Chemin de « data checkpoint » du puits S_m[40]

2.2.1.3. Redondance hétérogène

Une autre approche [41] utilisée dans les réseaux de capteurs introduit le concept de la redondance hétérogène. Un capteur en panne peut être remplacé par un capteur d'un autre type. Le remplacement d'une ressource par une autre nécessite une adaptation adéquate au niveau système et au niveau application. Prenons l'exemple d'un réseau de capteurs pour la reconnaissance de personnes déployé dans une société pour identifier ses employés. Six personnes nommées A, B, C, D, E et F travaillent dans cette société. Le système de reconnaissance utilise deux types différents de capteurs : 1) capteur de taille (grandeur) ; 2) capteur de reconnaissance de la voix qui demande à chaque entrant d'introduire une phrase secrète donnée à l'aide d'un microphone. La Figure 2-3 montre les six personnes ainsi que leurs caractéristiques (taille et voix) représentées dans le graphe.

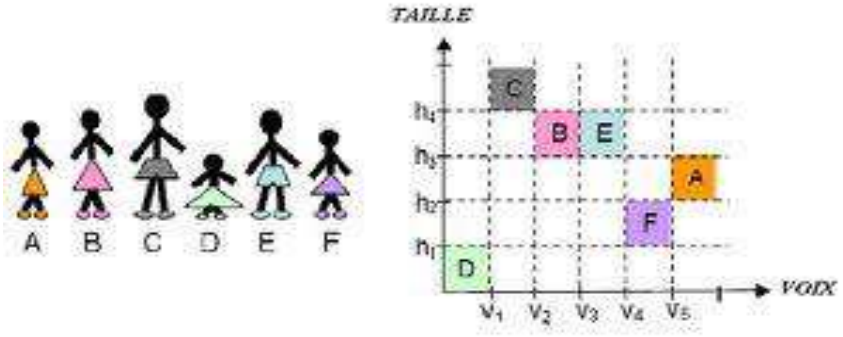


Figure 2-3: Exemple d'un RCSF tolérant aux pannes [42]

Dans cet exemple, si tous les capteurs fonctionnent correctement, chaque personne occupera une surface différente. En outre, dans la plupart des cas, et malgré la défaillance de l'un des capteurs de taille ou de voix, la reconnaissance de toutes les personnes est encore possible. Cependant, pour le cas des personnes B et E, qui ont la même taille, la voix est le seul critère pour les distinguer. Par conséquent, le système ne peut tolérer aucune panne du capteur V3 qui distingue B et E. Si on exclut B ou E du personnel de la société, alors le système sera complètement tolérant aux pannes. L'approche est avantageuse du point de vue de l'autonomie du système parce que la solution se limite aux ressources propres du système. Cependant, elle dépend de l'application et des types des capteurs utilisés.

2.2.1.4. Routage des chemins

Le projet PEQ [43] combine la conservation d'énergie avec le routage multi-chemins en sélectionnant parmi toutes les routes disponibles, celles qui consomment le moins d'énergie. En plus de ce mécanisme préventif qui permet un

routage fiable, un mécanisme de recouvrement de pannes est implémenté. Ce dernier remplace le chemin en panne par une autre route faite de liens fiables et qui consomme moins d'énergie. Ainsi, le protocole PEQ assure la procédure de tolérance aux pannes par la gestion de la consommation d'énergie, la sélection des meilleures routes puis leur recouvrement en cas de panne. Fa2TD [44] offre une solution curative de détection et de recouvrement de pannes dans le protocole « Directed Diffusion » [45]. Il offre un meilleur temps de reprise après panne et réduit la durée d'inactivité du réseau ainsi que le nombre de paquets perdus. Pour ce faire, il définit un paramètre qui associe à chaque chemin renforcé un délai de garde pour détecter les éventuelles pannes durant la transmission des données. Si le délai de garde expire et aucune donnée n'est reçue, FaT2D déclenche un événement de détection de panne. Suite à cette notification, FaT2D diffuse une demande d'exploration afin de trouver un nouveau chemin fiable remplaçant le chemin défaillant

2.2.1.5. Redondance temporelle

Le projet FATE_CSQ [46] offre une solution tolérante aux pannes pour l'évaluation des requêtes continues de type sélection dans les réseaux de capteurs sans fils. Il vise à prévenir les pannes par une réorganisation occasionnelle de la topologie du réseau, à détecter les pannes par un des mécanismes d'acquittements intelligents avec « feedback » et à les traiter par des mécanismes de retransmission des réponses perdues. FATE_CSQ garantit à l'utilisateur un niveau de qualité spécifié par la requête lancée. Si l'utilisateur a lancé une requête demandant « s'il y a une fuite de gaz toxique » dans un réseau de 1000 capteurs et s'il a reçu 100 réponses positives à sa requête, l'utilisateur ne connaît pas ce qu'il se passe pour les 900 capteurs restants. Est-ce qu'ils ont une réponse négative à la requête (pas de fuite) ? Ou leurs réponses sont perdues (mais il y a une fuite) ? Si l'utilisateur peut savoir que 400 capteurs envoient des réponses négatives, alors il obtiendra une meilleure précision sur la réalité parce qu'il a reçu 100 réponses positives parmi 600 réponses potentielles et non pas parmi 1000. Le niveau de qualité est mesuré par le rapport suivant :

$$= \frac{|A|}{|S| - N}$$

$$Q = \frac{|A|}{|S| - |N|} \qquad (2.2)$$

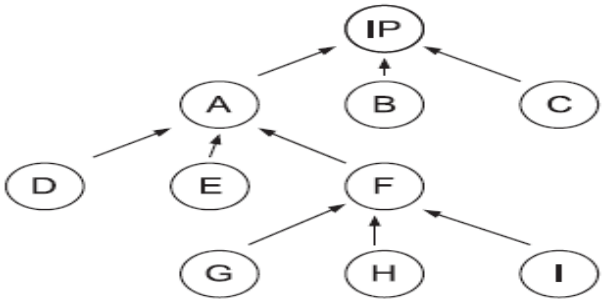


Figure 2-4: Arbre de routage [46]

|A| : nombre des réponses positives, |S| : nombre total des capteurs, N : nombre des réponses négatives. Considérons l'exemple suivant : |S|=1000 capteurs et |A|=100 et N=500 alors q=20%. L'utilisateur a reçu seulement 20% des réponses potentielles à cause des défaillances. La connaissance du rapport de garantie aide l'utilisateur à prendre des décisions plus précises. L'algorithme doit compter alors le nombre des réponses négatives pour satisfaire la requête de l'utilisateur.

L'algorithme commence par la diffusion de la requête à travers le réseau permettant de construire un arbre de routage. Chaque nœud maintient l'ID de son parent et les ID de ses enfants. Puis, chaque nœud envoie à son parent sa réponse s'il est positif, sinon il envoie un message de négation. Si le nœud a un sous-arbre, il renvoie les messages positifs et le nombre de messages de négation de ses nœuds fils. Le processus d'évaluation se répète jusqu'à la fin de la période ou la satisfaction de l'exigence de qualité. A la fin de chaque tour, un nœud parent diffuse un vecteur binaire d'acquittements à son sous-arbre. Chaque nœud fils vérifie le vecteur et retransmet la réponse en cas de perte (bit 0). C'est ce qu'on appelle « *Direct Feedback* » entre le nœud parent et les nœuds fils directs.

De même, pour la perte des réponses provenant des descendants des nœuds fils, par exemple le cas où la réponse du nœud H (Figure 2-4) se perd sur le lien F-A, l'algorithme introduit un autre mécanisme d'acquittement appelé « *Forwarding Feedback* ». Le but de ce mécanisme est de retransmettre les réponses perdues provenant des nœuds plus lointains que le nœud fils direct. La retransmission se fait du nœud F et non pas du nœud d'origine H, ce qui réduit le temps de latence et la surcharge de communication. La topologie du réseau peut subir, pendant la phase d'évaluation de la requête, des modifications dues à la coupure des liens et/ou défaillances des nœuds dans le réseau. C'est pourquoi un nœud parent peut décider de déclencher la phase de restructuration de la topologie avant de commencer la période suivante. Il diffuse un message « Restructure » permettant de rétablir la relation entre un nœud parent et les nœuds descendants.

Le projet FATE-CSQ rationalise la consommation d'énergie de plusieurs façons :

- Il évite la collision lors de la transmission : FATE-CSQ utilise *le mécanisme TDMA* (couche MAC) pour allouer le temps de transmission pour chaque nœud fils, ce qui évite la communication simultanée des nœuds fils ;
- Il profite de la nature de communication dans les réseaux sans fils. En effet, lorsqu'un capteur envoie un message, celui-ci sera écouté par l'ensemble des capteurs qui sont dans la même portée de la communication. Ainsi, le feedback est envoyé dans un seul message pour tous les fils ;
- La réponse d'un fils est transmise à son parent au plus une seule fois ;
- Il réduit au maximum le temps d'*écoute passive* en transmettant le feedback directement à la fin de la transmission des réponses ;

- Le mécanisme de fiabilité (Feedback MTV) est du type *saut par saut* et non de bout en bout ; ce qui permet de retransmettre le message à partir du lien où il est perdu et non pas à partir de l'émetteur d'origine ;
- Le capteur entre dans l'état de sommeil lorsqu'il a accompli son rôle de transmission (envoie sa réponse et les réponses de ses fils à son parent) avant la fin de la période de transmission.

Par contre, la phase de reconstruction de l'arbre introduit une surcharge considérable due aux messages de diffusion transmis.
Dans la suite, nous allons decrire des solutions fondées sur la détection et le diagnostic des fautes. Des telles solutions permettent de minimiser le recours à la redondance et prend alors en compte les aspects liés à l'autonomie et les contraintes des ressources imposées par la nature du RCSF.

2.2.2. Détection des fautes des données

Les données de capteurs sont parfois incomplètes et bruitées, parfois redondantes. Ces données "sales" doivent être "nettoyées" par des techniques de "data cleaning" [47]. Cependant, contrairement aux techniques traditionnelles de nettoyage des données pour les entrepôts de données, le nettoyage des données de capteurs doit être fait en temps réel, soit au niveau capteur [48] soit au niveau puits [49].

2.2.2.1. Origines des fautes

Les erreurs dans les données peuvent surgir à différents niveaux du système de RCSF. Ces erreurs sont directement liées aux processus d'acquisition et de traitement des données [8]. Elles représentent des défaillances spécifiques aux quatre facteurs principaux suivants : le capteur, les méthodes de mesures, le facteur environnemental, et la communication.

- *Le capteur :* Le capteur est un dispositif qui transforme les signaux physiques du monde réel en données. Ces appareils ont des capacités pour mesurer, calculer et communiquer les mesures acquises. Nous avons deux aspects importants à considérer : l'état physique du nœud/capteur et la limitation des ressources. Concernant l'*état physique du nœud* capteur nous considérons que le temps de vie du capteur, son calibrage, la dégradation de l'appareil, le mauvais fonctionnement, ou les erreurs d'installation *etc*., sont des aspects qui peuvent impacter la qualité des données lors de la prise de mesure. L*es ressources du capteur* représentent aussi une source d'erreurs. Le fait que les capteurs sont typiquement alimentés par des batteries (souvent non renouvelables), et le changement de ces batteries qui est souvent très délicat, voire impossible selon la localisation du capteur, le milieu d'observation et le type de réseau utilisé, font que la probabilité qu'ils manquent à des tâches prévues n'est pas négligeable. De plus, les limitations des capacités de calcul et stockage réduisent la capacité et la quantité de traitement exécuté au sein des capteurs. Ceci peut générer une perte d'information, de mauvais calculs, l'indisponibilité du capteur, ou encore l'envoi de données au serveur central sans demande directe.

- *Les méthodes de mesure :* Généralement, les capteurs mesurent un seul élément (i.e. humidité, température…) à la fois, en considérant certaines exceptions (i.e. capteur RDI – pression, champ magnétique, température). Ceci est du au fait que la taille, l'énergie et la capacité de traitement au sein du capteur sont limitées. Ces aspects incitent à utiliser des méthodes de mesure peu coûteuses en termes de capacités (activation par périodes de temps, l'échantillonnage, l'activation par détection des seuils…) en dépit de la fiabilité de la donnée en termes de précision et exactitude de résultats.

- *Le facteur environnemental :* Les réseaux de capteurs sont souvent déployés dans des zones à risque, donc les conditions de fonctionnement d'un capteur ou d'un réseau de capteur peuvent être affectées par les conditions environnementales (i.e. inondations, neige, orage, séismes, *etc*.). De même, les conditions de fonctionnement des capteurs peuvent ne pas être optimales (i.e. températures élevées, excès d'humidité, *etc*.). Ces facteurs ont un impact important sur la qualité du fonctionnement des capteurs, et par conséquent sur les mesures qu'ils réalisent.

- *Communication :* Une fois les données collectées (acquises et validées), les capteurs/nœuds capteur utilisent souvent une communication sans-fil pour communiquer les données acquises à un nœud central ou à un serveur. Cependant, cette communication peut être endommagée à cause de mauvaises conditions environnementales, le partage des ressources du capteur *(i*nterférence) ou le partage d'une même chaine de communication avec des autres capteurs au sein du réseau (congestion), *etc*. Une mauvaise communication peut générer des données erronées, données manquantes, des données en retard ou l'indisponibilité des sources. Ces aspects impacteront de façon importante la qualité de l'analyse des données en temps réel.

2.2.2.2. Exemples d'approches de détection

A. Test de cohérence des données

Dans [50], un algorithme de test en ligne est proposé pour détecter les capteurs qui donnent des mesures incorrectes. L'idée essentielle consiste à analyser l'effet d'un capteur particulier sur la cohérence des résultats obtenus par la fusion de plusieurs capteurs. Si l'élimination d'un capteur améliore la cohérence des résultats, alors le capteur sera considéré comme défectueux. L'algorithme consiste à appliquer le processus de la fusion n+1 fois (où n est le nombre de capteurs). A chaque itération, on élimine un capteur et on réapplique le processus de la fusion afin d'identifier son impact sur la cohérence des valeurs. Après n+1 itérations, on arrive à identifier le capteur le plus discordant. L'étude expérimentale montre que l'algorithme permet de détecter 100% des capteurs défectueux lorsque les mesures des capteurs sont altérées de 7%.

B. Corrélation spatiale et temporelle

Dans [51][52][53][54], l'idée consiste à comparer les mesures de capteurs similaires qui se trouvent dans une même zone géographique. Tant que les valeurs délivrées par ces capteurs restent égales entre elles, l'information est considérée comme fiable car il est très improbable que tous les capteurs fassent la même erreur de mesure au même instant. Si une de ces valeurs s'écarte significativement des autres, c'est qu'un problème est apparu sur le capteur qui délivre cette valeur aberrante. Dans [52], l'algorithme identifie tout d'abord les capteurs qui sont en bon état, ce après une suite d'échanges de valeurs, puis utilise ces capteurs comme références pour tester les autres capteurs. La simulation montre que l'algorithme a un taux de faux négatifs assez élevé, soit de 97%, en présence de 25% de capteurs défectueux. Cependant, il induit une surcharge assez élevée en termes de communication et d'énergie. Pour réduire la surcharge du réseau, une architecture arborescente est adoptée dans [54]. Après un certain nombre d'échanges, un capteur est considéré en bon état. Il sert alors de référence pour tester les capteurs de son sous-arbre.

2.2.3. Détection et diagnostic des fautes au niveau nœud/réseau

Le diagnostic est défini comme étant l'identification de la cause probable de la (ou des) défaillances à l'aide d'un raisonnement logique fondé sur un ensemble d'informations provenant d'une inspection ou d'un contrôle sur le système à diagnostiquer. Par exemple, le système peut observer une perte de données et la diagnostiquer en identifiant si elle provient de fautes logicielles, de pannes de nœuds, de problèmes de liaisons, *etc*.

2.2.3.1. Procédure de diagnostic

Le diagnostic s'effectue en trois étapes : l'observation des symptômes, l'interprétation des symptômes et l'identification du composant.

- *Observation des symptômes*. Un symptôme est un signe comportant une information révélatrice de l'état réel du système. Cette étape nécessite la collecte d'informations sur l'état du système afin de le diagnostiquer. La collecte d'informations est une étape critique dans la conception du système. Une expertise dans la sélection des métriques et le choix du mécanisme d'extraction et d'échange dans le réseau permettent une maîtrise de la consommation énergétique.

- *Interprétation des symptômes.* C'est la phase de détection de la défaillance dans laquelle on reconnaît qu'un événement inattendu s'est produit. Les techniques de détection de pannes sont généralement classifiées en deux

catégories, en ligne et autonome (offline). La détection offline est souvent réalisée à l'aide de programmes de diagnostic qui s'exécutent quand le système est inactif. La détection en ligne vise l'identification de pannes en temps réel et est effectuée simultanément avec l'activité du système. La détection efficace de la défaillance doit réduire le plus possible les taux de *faux positifs* (fausses alertes) et *négatifs* (le taux de non détection en présence de défaillance). Un taux élevé de faux négatifs dégrade la performance du système. Un taux élevé de faux positifs augmente inutilement les coûts de maintenance.

– *Identification du composant* (ou ensemble) défaillant : C'est la phase du diagnostic de la défaillance : Elle a pour but de remonter à l'origine de la défaillance détectée. L'identification du composant défaillant nécessite une analyse des dépendances entre les causes et les symptômes observés. Une analyse rigoureuse des informations, basée sur un raisonnement logique, permet d'identifier les causes principales des défaillances. La connaissance des causes principales de la défaillance améliore l'efficacité des opérations de *maintenance.*

2.2.3.2. Critères de classification

De nombreuses approches ont été développées par différentes communautés de recherche, en vue du diagnostic des défaillances, dans le système à base de RCSF. Elles se distinguent par les mécanismes de la collecte les informations du diagnostic, la structure de la prise de décision concernant la détection, et les techniques d'analyse les symptômes observés (Figure 2-5)

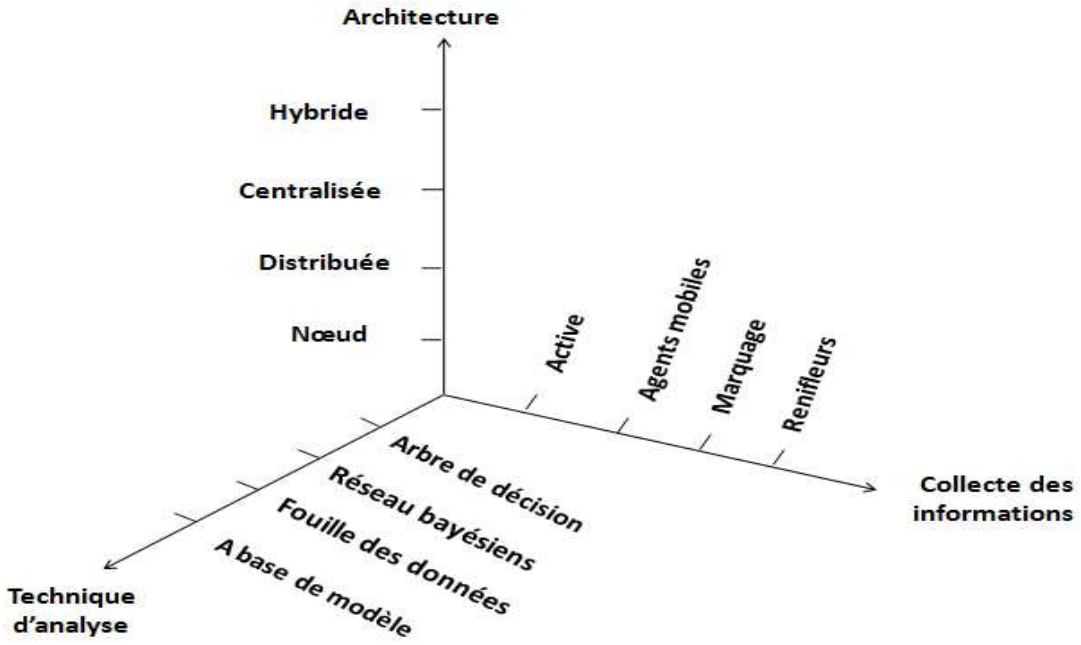


Figure 2-5: Critères de classification des approches de diagnostic

A. Mécanismes de la collecte d'informations

L'efficacité d'un système de diagnostic dépend grandement de la pertinence des informations recueillies sur le système à diagnostiquer. Si cette information n'est pas suffisante, le système de diagnostic produira des informations peu précises.

- *Types d'informations*. Les informations du diagnostic peuvent être de nature différente : métriques systèmes [55][56] (le taux d'occupation du processeur, le taux d'occupation mémoire, *etc*.), le trafic des paquets [58][59], des variables [60][61], les traces d'événements [62][63], des résultats d'évaluation des prédicats et assertions [64][65], mesures de puissance[66]. L'outil de diagnostic doit sélectionner prudemment les informations favorables aux modèles de fautes qu'il analyse, afin de rationnaliser l'utilisation des ressources.

- *Méthodes de la récupération d'informations* **:** l'outil de diagnostic récupère les informations sur le système surveillé selon deux approches principales [67] : active et/ou passive.
  - Approche active : Les nœuds génèrent des paquets spécifiques au diagnostic (les paquets de contrôle) (Figure 2-6). La transmission des paquets s'effectue en utilisant le canal de communication principal de l'application au niveau du nœud. L'approche active est coûteuse en termes de consommation de ressources. Pour cela, certaines solutions introduisent des nœuds « mobiles » dotés de batteries plus puissantes et capables de se déplacer pour récupérer les informations des nœuds du réseau. Cependant, elle ne nécessite pas de matériel supplémentaire pour transmettre les paquets de contrôle. Quant à la pile de protocole, la majorité d'approches utilise celle de l'application, tandis que d'autres introduisent leur propre pile protocolaire [68]. La dernière solution garantit un certain niveau de fiabilité au détriment du coût mémoire.

  - Approche passive : L'approche passive consiste à capter les flux de paquets par des unités de renifleurs. L'approche passive ne requiert aucune intervention de la part des nœuds du réseau, et ne génère pas de paquets supplémentaires dans le réseau. Elle peut être réalisée par le puits [55], les nœuds du réseau [69] ou par un système dédié de nœuds renifleurs indépendant du système surveillé [70][72]. Un avantage principal du diagnostic passif est la transparence. Le diagnostic s'effectue sans aucune interférence avec les opérations normales du réseau. Ce qui permet de garder les ressources des nœuds du réseau et la performance globale du système. Un autre avantage est le support d'une grande diversité de plateformes qui n'affecte pas les traces des paquets. La limitation principale de l'approche, est la limitation de la visibilité du système surveillé aux flux circulant dans le réseau. Par conséquent, le diagnostic sera restreint aux défaillances qui se manifestent seulement sur les flux des données. Une autre limitation se trouve au niveau de la

mobilité, où la localisation des nœuds est conditionnée par la couverture par des nœuds renifleurs.

Un autre modèle de l'approche passive, est l'approche de marquage des paquets « piggybacking ». Il consiste à insérer les informations de diagnostic dans les paquets des données. Il permet de réduire la consommation de la bande passante, en évitant d'injecter des paquets supplémentaires dans le réseau. L'inconvénient principal du marquage des paquets, est que la taille des informations à insérer est limitée à la taille maximale des paquets.

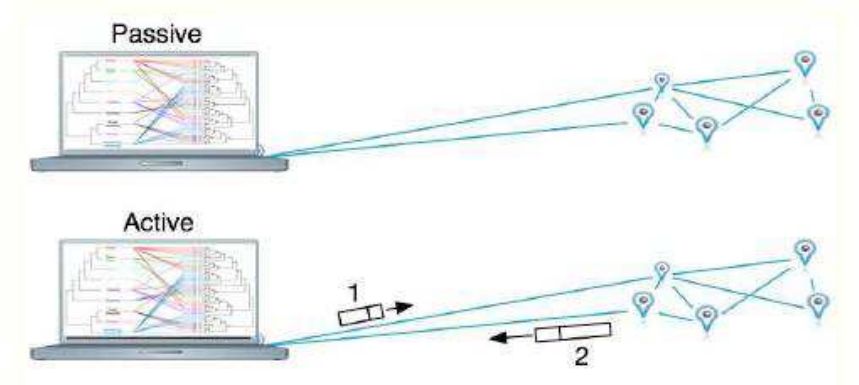


Figure 2-6: Méthode de la collecte d'informations de diagnostic [67]

– *Déclenchement de la collecte :* La collecte d'informations se fait selon une ou plusieurs stratégies : collecte proactive, collecte réactive, collecte à la demande. Pour chacune de ces stratégies, le nœud peut enregistrer temporairement les données sur un disque « flash ».
  - *Collecte proactive* [55] : les nœuds envoient les informations du diagnostic au(x) nœud(s) concerné(s) d'une manière périodique. La collecte active est une solution coûteuse en termes de consommation d'énergie. L'administrateur peut contrôler la consommation de la bande passante en augmentant la période de transmission d'informations. Dans ce cas là, un compromis entre la latence de détection et la consommation d'énergie doit être décidé en fonction des exigences de qualité des services attendus par le système.
  - *Collecte à la demande* [73] : Les nœuds envoient les informations du diagnostic à la demande de l'utilisateur du système.
  - *Collecte réactive* [74] : Elle consiste à appliquer des politiques, pour le déclenchement de la collecte des données, en fonction d'événements critiques observés (par exemple, observation d'un symptôme d'anomalie).

B. Structure de la prise de décision

Selon la structure de la prise de décision et le(s) entité(s) intervenante(s) dans le processus de détection et de diagnostic des défaillances, les auteurs en [75][76][77] classifient les approches de diagnostic en plusieurs catégories :

approche centralisée, approche distribuée, approche hybride et approche au niveau du nœud en [77] (Tableau 2-2).

- *Approche centralisée* : Elle repose sur un nœud central qui diagnostique les fautes en se basant sur des informations recueillies des nœuds du réseau. Cette approche assure une supervision globale de l'état du réseau permettant de garantir une exactitude quant au diagnostic des problèmes complexes (par exemple, la défaillance des nœuds critiques, l'occurrence des fautes simultanées ou l'affection d'une zone complète des nœuds, *etc*.). Les limites de cette approche sont liées à une dépendance forte due à la centralisation exclusive des opérations de diagnostic. D'une part, cette approche n'est pas robuste vis-à-vis de la perte des messages transmis sur un réseau sans fils de communication multi saut. De plus, l'arrêt accidentel du point central entraîne également l'arrêt du système de diagnostic. D'autre part, elle est coûteuse en termes de consommation d'énergie, et ne permet pas le passage à l'échelle. L'approche centralisée favorise principalement les réseaux de petite taille et les applications à base d'événements.
- *Approche distribuée* **:** La détection des fautes est réalisée de façon distribuée par l'ensemble des nœuds du réseau. Elles permettent de limiter la surcharge de la communication et d'améliorer la robustesse du système. Cependant, des mécanismes de coopération doivent être mis en œuvre pour assurer la cohérence dans les opérations de gestion exécutées par les nœuds.

Approche de la collecte d'information	Active	Passive	Marquage
Transparence	Non	Oui	Oui
Coût mémoire	Oui	Non	Non
Consommation d'énergie	Forte	----------	Faible
Fiabilité	Faible	élevée	Faible
Autonomie des ressources	Oui	Non	Oui
Domaine des défaillances	Illimitée	Limitée	Limitée
Support des plateformes variées	Non	Oui	Non
Support de la mobilité	Oui	Non	Oui
Passage a l'échelle	------	Oui	Oui
Domaines d'applications	Tout type d'application	Application de la collecte des données	Application de la collecte des données
Instrumentation du code	Oui	Non	Oui

Tableau 2-1 : Mécanismes de transmission des informations du diagnostic

- *Approche hybride* **:** Elle combine le principe des deux approches centralisée et distribuée. Le but est de pouvoir profiter des avantages des deux approches : l'exactitude et la précision des approches centralisées et l'efficacité de la gestion de l'énergie et le passage à l'échelle des approches distribuées.
- *Approche de diagnostic au niveau du nœud :* Le diagnostic se base sur des informations sur l'état d'un nœud sans tenir compte du comportement des autres nœuds du réseau. Cette approche ne permet pas le diagnostic des fautes réseaux.

Architecture	Centralisée	Distribuée	Hybride
Précision/Exactitude	Bonne	Moyenne	Bonne
Consommation d'énergie	Forte	Faible	Moyenne
Contraintes ressources sur les nœuds	Non	Oui	Moyenne
Passage a l'échelle	Non	Oui	Oui
Robustesse	Non	Oui	Oui
Latence de détection	Bonne	Faible	Faible
Impacts sur la performance du réseau	Oui	Faible ou moyenne	Moyenne
Complexité d'implémentation	Non	Oui	Oui
Visibilité limitée de l'état du système	Non	Oui	Non

Tableau 2-2 Comparaison entre les approches centralisées, distribuées et hybrides

C. Techniques d'analyse

Le diagnostic des fautes repose sur des techniques d'analyse différentes, parmi lesquelles :

- *Diagnostic à base d'arbre de décision*[55][70] ***:*** Les méthodes à base d'arbre de décision ou à base des règles consistent à construire un arbre à questions successives (Figure 2-7). Selon la réponse, cet arbre peut être construit, et il permet de réaliser le diagnostic. Potentiellement, il est possible de diagnostiquer des fautes multiples en garantissant les diagnostics obtenus. Néanmoins, la garantie requiert que l'arbre de décision soit complet, et que toutes les situations soient représentées. En outre, il est nécessaire d'identifier *a priori* les fautes pouvant apparaître, à un certain moment, dans le système supervisé.

- *Méthode de diagnostic à base de modèle*[66][69] *:* Le diagnostic à base de modèles s'appuie uniquement sur la vérification de la cohérence entre le comportement réellement observé du système et le comportement attendu de ce système.

- *Méthode de diagnostic à base de réseaux bayésiens*[78] : les réseaux bayésiens sont la combinaison des approches probabilistes et de la théorie des graphes. Ils permettent de construire et développer des systèmes dynamiques évolutifs. Cette approche statistique requiert la construction de nombreuses probabilités, qu'il n'est pas aisé de régler. Le contexte probabiliste conduit à une absence de garantie : fondamentalement, tout est possible avec des probabilités plus ou moins importantes.

- *Méthode de diagnostic à base de fouille des données[79*] : Il s'agit d'enregistrer dans une base de connaissances les effets observés des défauts. Puis, lorsque des faits anormaux se produisent, il s'agit de rechercher des cas similaires dans la base de connaissances pour trouver des diagnostics possibles.

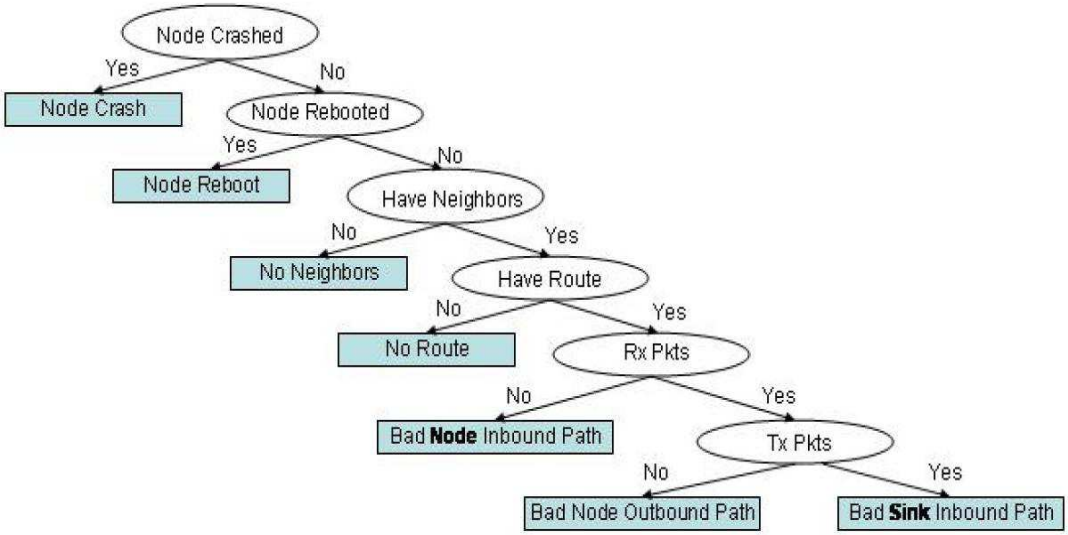


Figure 2-7: Diagnostic à base d'arbre décisions [55]

2.2.3.3. Exemples d'approches de diagnostic des fautes au niveau du nœud/réseau

Le diagnostic est défini comme étant l'identification de la cause probable de la (ou des) défaillances à l'aide d'un raisonnement logique fondé sur un ensemble d'informations provenant d'une inspection ou d'un contrôle sur le système à diagnostiquer. Dans la suite, nous allons décrire quelques exemples proposés dans la littérature.

A. Outil centralisé, actif, à base d'arbre de décision

Sympathy [55] est un outil de diagnostic centralisé (open source), conçu principalement pour les applications de collecte des données. Chaque nœud du réseau envoie périodiquement au puits les métriques définies par Sympathy. Celles-ci sont récupérées à partir de toutes les couches protocolaires du nœud, et elles sont de trois types : métriques de connectivité (table de routage, liste des voisins), métriques de flux (le nombre des paquets transmis et reçus par chaque nœud, le nombre des paquets échangés entre chaque nœud et le puits, l'estampille du dernier paquet reçu par le puits pour chaque nœud) et les métriques du nœud (temps de démarrage, le nombre de mauvais et de bons paquets reçus). Le puits collecte les métriques de manière passive en écoutant le trafic écoulé dans le réseau et de manière active car les nœuds envoient explicitement les métriques à chaque période T. La collecte n'est possible que pour les nœuds qui sont au prochain saut du puits. Pour cette raison, la collecte active des métriques est une obligation même si elle est très coûteuse en termes de communication et d'énergie. Une fois les métriques collectées, le puits peut identifier le type de(s) panne(s) produite(s) (franche, omission, synchronisation, transmission), ses causes principales (l'écrasement d'un capteur, le redémarrage du puits, l'absence de voisins, l'absence de route au puits, un mauvais chemin vers le nœud, un mauvais chemin au puits) et ses sources (le nœud lui-même, le réseau ou le puits).

De fait, une panne est détectée quand un capteur (ou un composant d'une application) génère un trafic (métrique flux) inférieur à celui prévu. Ensuite, un arbre de décision est conçu pour faciliter l'analyse des causes des pannes (panne franche du nœud, perte de connectivité, problèmes dans le routage, *etc*.). Par exemple, si le puits n'a pas reçu un paquet d'un nœud N au bout d'une durée donnée (après laquelle les métriques deviennent invalides) et si le nœud N n'est inclus dans la liste des voisins d'aucun autre nœud du réseau, N est supposé écrasé (panne franche du nœud). Par la suite, « Sympathy » archive les pannes afin que l'utilisateur puisse effectuer les étapes de reprise convenablement. Comme toute approche centralisée, « Sympathy » induit une surcharge dans la communication (surcharge de 30% du trafic écoulé [70]), ce qui limite le passage à l'échelle. De plus, il nécessite l'instrumentation du code qui le rend vulnérable par rapport aux bugs logiciels du capteur.

B. Outil centralisé, marquage des paquets, à base de modèle d'inférence probabiliste

PAD [78] est une approche centralisée à base de modèle d'inférence, conçue pour les applications de collecte des données. Elle collecte les informations de diagnostic par le marquage des paquets. PAD est constitué de quatre composants essentiels :

- Un module de marquage réside à chaque nœud du réseau. Il sert à insérer les champs de diagnostic dans les paquets des données : son identifiant, l'identifiant du nœud source et le numéro de séquence du paquet. Un paquet traversé dans le réseau ne peut être marqué que par un seul nœud choisi en fonction d'un ensemble des règles définies par l'algorithme (Figure 2-8)
- Un module d'analyse des marques réside au niveau du puits. Il analyse les marques, génère la topologie du réseau, détecte des symptômes anormaux (perte, retard, ou duplication des paquets), génère des diagnostics préliminaires (les liens qui provoquent une perte des paquets, modification fréquente du nœud parent, les liens et les nœuds qui sont en bon état à cause de la bonne réception. *etc*.).
- Un modèle d'inférence probabiliste construit les graphes de dépendances entre les éléments du réseau.
- Un moteur d'inférence. Il se base sur un réseau bayésien pour identifier les causes principales des symptômes anormaux. . Il produit les diagnostics finals, en se basant sur le modèle d'inférence et les symptômes (négatifs et positifs) générés par le système de marquage.

Les avantages principaux de cette approche sont la construction dynamique des modèles de fautes et la topologie du réseau, la prédiction de certains types de fautes et d'informations sur l'état du réseau, la gestion efficace de l'énergie et le passage à l'échelle. La gestion de l'énergie est effectuée de manière efficace par

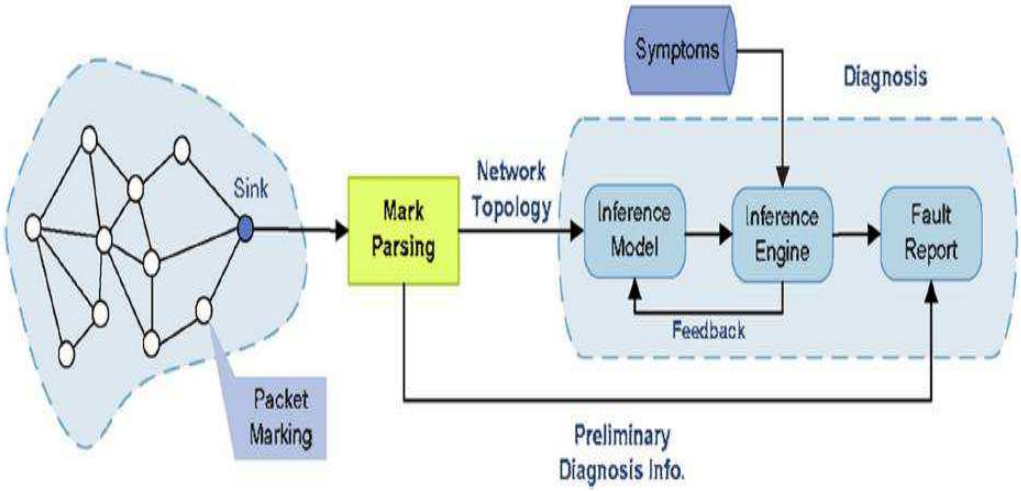


Figure 2-8: PAD Aperçu général du système [78]

l'analyse passive des symptômes observés et par la minimisation de la quantité des informations échangées (2 octects). Cette minimisation est possible grâce au système d'inférence probabiliste qui permet de compléter les informations de diagnostic par déduction des états internes des éléments du réseau.

PAD montre une efficacité à identifier les causes des problèmes observés (déplétion rapide de l'énergie, perte des données et retard) dans le cadre du projet « OceanSence » [81]. PAD identifie les fautes de type destruction physique, écrasement de logiciel, fautes au niveau de l'application, congestion et interférence. Les modèles des fautes sont construites en se basant sur les connaissances d'experts de la nature des fautes ou une déduction à partir des données historiques du réseau

C. Outil centralisé, actif, à base de corrélation des métriques

AD [56] présente une approche centralisée et agnostique de diagnostic des fautes. Cette approche n'exige pas une connaissance *a priori* de l'environnement et des types de fautes qui peuvent apparaître pendant le déploiement du réseau. AD collecte périodiquement vingt deux métriques qu'on peut classifier en quatre catégories : métriques de chronométrage, métriques de trafic, métriques concernant les tâches du système et métriques de connectivité (par exemple, le temps cumulé, le nombre de paquets transmis par un nœud, le nombre de tâches exécutées, le nombre de changements du nœud parent, *etc*.). AD analyse la corrélation temporelle et/ou spatiale des métriques pour détecter les fautes produites (Figure 2-9). Pour cela, à chaque période T, les corrélations de chaque nœud sont calculées et représentées par une matrice dont chaque élément correspond à une corrélation entre chaque couple de métriques. Un changement significatif entre deux mesures consécutives des corrélations d'un nœud indique la présence d'une faute. En plus de la corrélation temporelle, AD détecte des fautes par la corrélation spatiale entre les nœuds qui ont des fonctionnalités similaires, par exemple les nœuds feuilles du réseau.

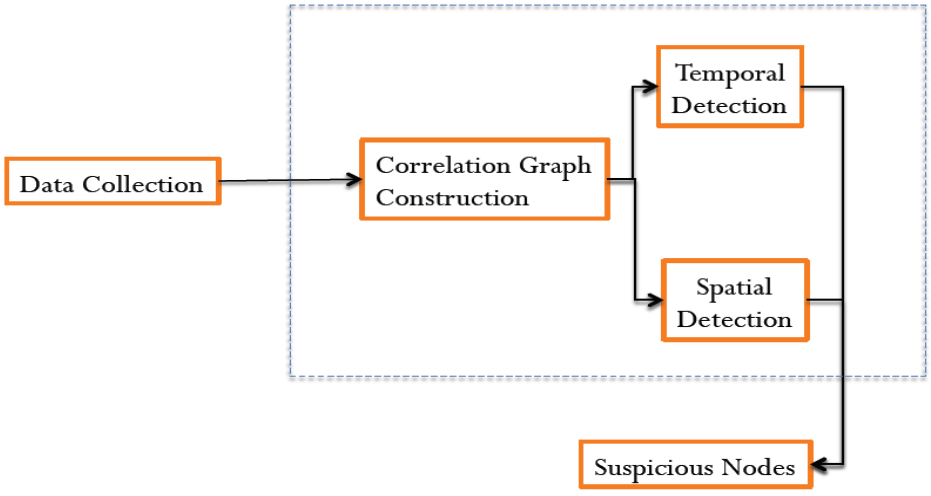


Figure 2-9: Approche agnostique de diagnostic [56]

AD montre une efficacité de détecter des fautes potentielles (« ingress drops », problèmes de routage, problèmes de liens, fautes logicielles) dans le cadre du projet « GreenObs » [57]. Certaines de ces fautes peuvent être détectées par l'observation directe des valeurs anormales de certaines métriques, tandis que d'autres sont détectées par l'observation de la corrélation de certaines métriques. Par exemple, le nombre de paquets transmis par un nœud correct doit être proche du nombre de paquets reçus par les nœuds descendants. AD permet de réduire le taux de faux positifs et négatifs avec une surcharge d'énergie raisonnable où l'ensemble des métriques sont envoyées périodiquement à un nœud central dans un seul paquet.

D. Outil centralisé, réseau dédié, à base de modèle

PowerTracer [66] introduit un système de diagnostic qui se base sur un modèle de la consommation d'énergie des nœuds. PowerTracer crée des modèles des fautes (modèles de référence) par leur émulation lors d'expériences en laboratoire et par l'utilisation des techniques d'apprentissage (le modèle de chaîne de Markov). Le processus de diagnostic peut être effectué, en quelques minutes, en comparant les modèles de référence aux modèles observés. Ces derniers sont construits selon une fenêtre temporelle, par exemple d'une durée de 30 min, à partir des mesures collectées préalablement du nœud en panne. L'exactitude de l'outil de diagnostic dépend de la taille de l'espace des états de transition de la chaîne de Markov, de l'intervalle d'échantillonnage, et de la taille de la fenêtre temporelle.

L'étude expérimentale montre que l'identification des fautes peut être exacte à 100% avec un choix convenable de trois facteurs précédents. Cependant la mise en œuvre de PowerTracer nécessite d'installer trois types de composants : un mètre de puissance pour chaque nœud du systèmeet une station de diagnostic qui analyse les traces de puissance collectées à travers des liaisons sans fils de faible bande passante. Ce coût supplémentaire (estimé à 3%) est acceptable par rapport à l'indépendance réalisée du système de diagnostic par rapport au système de surveillé.

E. Outil hybride à base d'analyse de flux des données

PD2[82] est un outil de diagnostic hybride conçu principalement pour les applications de collecte des données. Il effectue une analyse de la performance des flux des données générées par les nœuds pour isoler les éléments (nœuds, composants matériels ou logiciels) potentiellement défectueux. Pour cela, PD2 identifie chaque flux de données par l'ensemble des composants qu'il traverse à partir du nœud source jusqu'au nœud destination, le puits. Il garde la trace des chemins par un ensemble de règles de dépendances qui relient les différents modules logiciels des nœuds traversés par les flux. Lorsqu'une faute apparaît et provoque un taux de perte et/ou un temps de latence au niveau d'un flux de données, PD2 commence à étudier le problème en parcourant le chemin en sens inverse à partir du nœud puits. Il identifie les éléments défaillants en calculant le taux de perte et le temps de latence de chaque composant du chemin. Une fois les localisations des fautes identifiées, il est possible d'extraire

des informations du composant défaillant et d'appliquer un raisonnement logique proposé par d'autres outils de débogage (par exemple « Sympathy » [55]) pour identifier les causes des problèmes. C'est de cette manière que PD2 arrive à contrôler la consommation d'énergie en collectant les informations de débogage seulement des éléments défectueux dès que la défaillance d'un élément affecte la performance de l'application.

F. Outil hybride, réseau de renifleurs, à base d'arbre de décision

SNIF [70] est un outil de diagnostic passif et à base d'arbre de décision pour les applications de collecte des données du RCSF. Il consiste à déployer un réseau de renifleurs distribués entre les nœuds capteurs. Les renifleurs écoutent passivement le trafic du réseau de capteurs pour éviter l'impact de l'instrumentation du code et l'envoi des messages explicites sur l'activité du réseau surveillé. Ils analysent le trafic pour extraire les indicateurs de fautes. SNIF peut identifier des fautes présentes au niveau des nœuds et du réseau. Cependant, certains types de fautes ne peuvent être identifiés directement par l'observation de certains indicateurs (par exemple, la défaillance d'un nœud suite à l'épuisement de l'énergie) ou déduites par l'analyse de plusieurs symptômes observés. Bien que L'approche SNIF induit un coût d'installation supplémentaire et nécessite une compréhension profonde des protocoles utilisés dans le réseau, elle convient à de nombreuses applications pour plusieurs raisons : elle laisse intactes les ressources du système, elle évite l'impact de l'instrumentation de code et de la modification du trafic du réseau et elle garantit la fiabilité du système de diagnostic. Ce dernier n'est pas affecté par les problèmes du réseau des capteurs produits pendant le déploiement.

Il est à noter que le trafic des paquets est une source importante d'informations pour de nombreuses approches de diagnostic à base de renifleurs. Les messages d'application renferment des champs de trois ou quatre couches de la pile protocolaire (Figure 2-10) :

- *La couche MAC* spécifie comment les données sont expédiées entre deux nœuds dans une distance d'un saut. Elle génère trois types de messages : « unicast », acquittement et diffusion. Chaque message contient l'adresse source (l'adresse du nœud émetteur) et l'adresse de destination (l'adresse du nœud récepteur). Lorsque le nœud destination reçoit avec succès un message « unicast », il envoie un accusé de réception, qui peut être utilisé par l'outil de diagnostic pour conclure que le destinataire a bien reçu le message (exemple de détection de la perte des messages).
- La *couche de synchronisation* temporelle (couche optionnelle). L'information extraite de cette couche permet de savoir si les horloges des nœuds sont bien synchronisées (problème de synchronisation).
- La *couche réseau* dont le but principal est de trouver une route pour la transmission des données, captées des nœuds capteurs vers le puits en optimisant l'utilisation de l'énergie des capteurs. Les informations fournies par cette couche concernent la qualité des liens entre un nœud capteur et ses voisins et la distance entre le puits et le nœud capteur. La distance peut être exprimée, par exemple, par

le calcul du nombre de sauts entre le puits et le nœud capteur. Cette information peut être utilisée pour déterminer le bon fonctionnement de la couche réseau (congestion, duplication des paquets, détection des zones, *etc.*).

- La couche application dont le rôle est d'assurer l'interface avec les applications. Il s'agit donc du niveau le plus proche des utilisateurs, géré directement par les logiciels. Les informations fournies par cette couche sont les lectures du capteur et autres informations spécifiques à l'application.

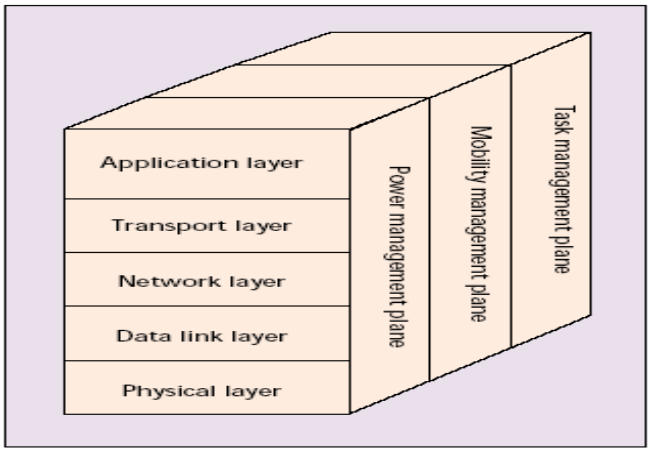


Figure 2-10: Pile protocolaire de RCSF [71]

G. Outil hiérarchique, agents mobiles, à base d'arbre de décision

L'approche proposée dans **[83]** se base sur un ensemble d'agents mobiles en charge de la détection intelligente des pannes et du recouvrement avec une gestion efficace de l'énergie et de la communication. L'architecture du réseau est hiérarchique à trois niveaux : les clusters, formés d'un ensemble de capteurs associés à un « ClusterHead », les « SuperClusters », formés de plusieurs clusters associés à une passerelle (Gateway) et le puits (Figure 2-11). Dans chaque niveau de la hiérarchie se trouvent deux types d'entités mobiles : les renifleurs (sniffers) et les correcteurs (correctors). Le renifleur émigre d'un nœud à un autre pour détecter les pannes et communique au(x) correcteur(s) leur localisation. La reconnaissance des pannes se fait grâce à une base de données distribuée. Cette base est définie partiellement au niveau du « ClusterHead » et des passerelles et elle est complète au niveau du « puits ». Elle décrit en détail les modèles de fautes et la relation qui existe entre les fautes, les erreurs et les défaillances.

Les interactions entre les agents mobiles sont inspirées du langage de la danse des abeilles. Celui-ci a l'avantage d'assurer la communication entre les abeilles (agents mobiles) en temps réel et avec une gestion efficace de l'énergie [84]. Les agents mobiles imitent les abeilles dans leur manière de communiquer pour indiquer la source et la qualité de nourriture (panne) en effectuant une danse en rond si la

nourriture est à moins de 15 mètres (message Time To Live TTL limité) du nid et une danse de transition pour la source à moins de 50 mètres.

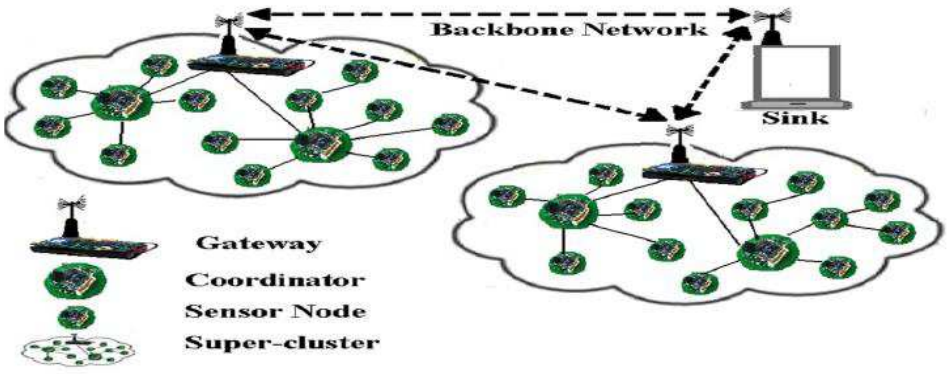


Figure 2-11: Architecture du réseau [82]

L'algorithme classifie alors les pannes en trois catégories selon leur gravité : la panne endémique, la panne épidémique et la panne pandémique. La panne est **endémique** si elle se produit à un taux faible au niveau d'un cluster. Par exemple, le débordement des tampons, le taux d'erreur binaire, *etc*. Ce type de pannes est traité par des *agents mobiles locaux* au niveau du cluster. La panne est **épidémique** si elle se produit à un taux plus élevé dans plusieurs clusters du même super cluster. Par exemple, la perte des paquets est considérée du type épidémique lorsqu'elle se produit dans plusieurs clusters et à un taux élevé. *Des agents métropolitains*, initialement déployés dans les passerelles, interviennent pour traiter ce type de panne. La panne est **pandémique** lorsqu'elle devient incontrôlable et se produit simultanément dans plusieurs super-clusters du réseau. Le traitement de ce type de panne se fait par des *agents globaux*

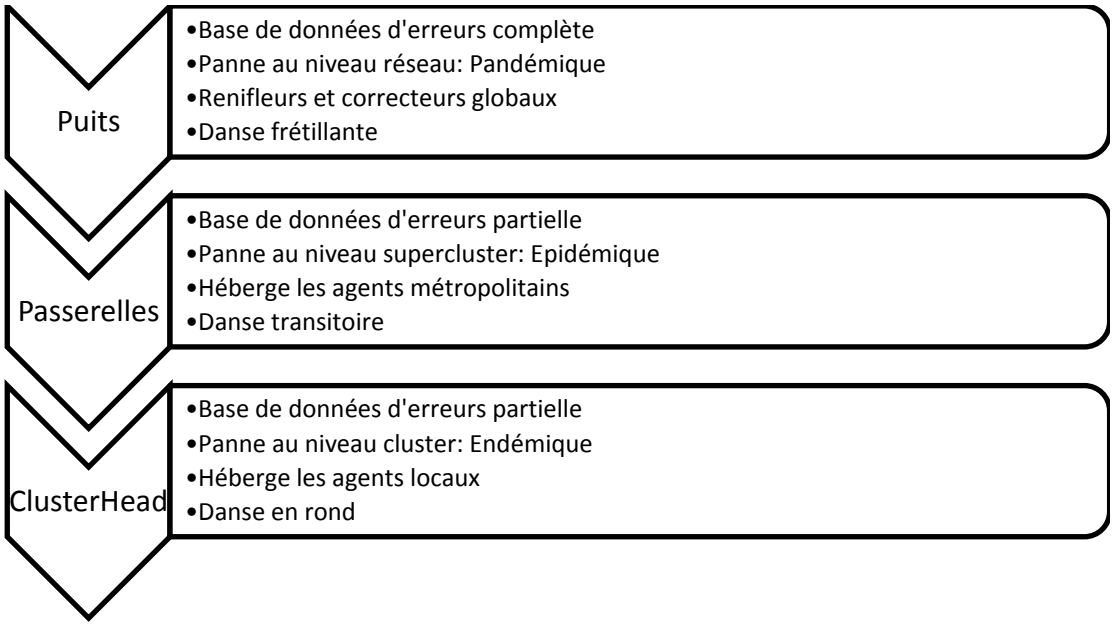


Figure 2-12 : Traitement hiérarchique des pannes

déployés dans le puits. La détection commence dans un cluster lorsque le renifleur local visite périodiquement les nœuds du cluster pour collecter et sauvegarder les paramètres pertinents à la détection des pannes (la table de routage, la liste des voisins, le temps de démarrage d'un nœud, le niveau d'énergie, *etc*.).

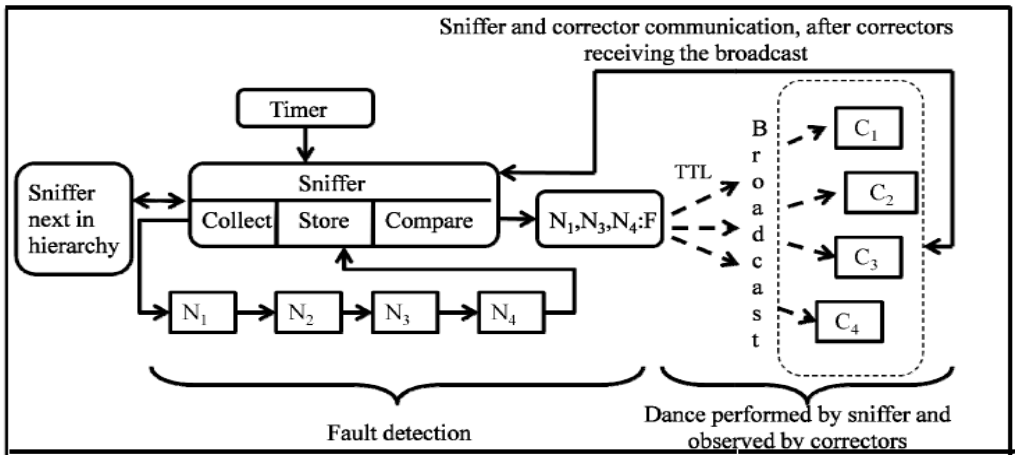


Figure 2-13 : Détecteurs des fautes [83]

Il détecte l'erreur lorsqu'il observe un comportement aberrant dans la plupart des paramètres ou découvre une déviation imprévue (dépassement d'un seuil prédéterminé) de la valeur d'un paramètre dans le temps. Par exemple, le niveau d'énergie d'un nœud à un temps T+t est inférieur à celui du temps t. Le traitement des pannes dépend de leur gravité et se passe d'une manière hiérarchique (Figure 2-12). S'il y a une seule défaillance, elle est traitée par le correcteur local. Sinon (défaillances multiples) le détecteur informe les correcteurs du cluster par la « Danse en Rond », diffusion de TTL limitée au cluster, qu'il y a une défaillance de type « endémique » en indiquant la localisation des nœuds affectés. Dans ce cas, les correcteurs visitent les nœuds défectueux pour les traiter. Si l'occurrence d'un même type de défaillance dépasse un seuil prédéterminé, le détecteur local annonce par une « danse de transition », diffusion de TTL limitée au super cluster, qu'il y a une panne du type « épidémique » en indiquant la distance et la direction du cluster affecté. Dans ce cas, les correcteurs des autres clusters se déplacent vers le cluster désigné et participent au traitement des pannes. Finalement, si le nombre de pannes explose dans plusieurs clusters d'un super cluster, le renifleur global du puits effectue la « danse frétillante», diffusion de TTL illimitée, indiquant qu'il y a une panne de type « pandémique » et demandant aux détecteurs de répliquer les correcteurs locaux et métropolitains. Une fois la « danse de transition » terminée, le renifleur global demande aux correcteurs répliqués de retourner à l'état normal. Dans la Figure 2-13, le renifleur visite les nœuds N1, N2, N3 et N4. Il collecte des paramètres explicitement (choisis par l'administrateur comme le temps de démarrage, le niveau d'énergie, *etc*.) ou implicitement (à travers les protocoles de communication comme la table de routage, la liste des voisins, *etc*.), il les sauvegarde et les compare aux paramètres collectés précédemment. En cas de panne(s), il informe les correcteurs pour entreprendre les actions appropriées. S'il n'arrive pas à détecter une faute liée à

une panne observée, il transmet la faute inconnue au renifleur du niveau supérieur de la hiérarchie. L'algorithme gère efficacement l'énergie, par :

- l'introduction des agents mobiles qui permet de déplacer la complexité du routage des nœuds capteurs vers les agents mobiles. Cependant la mobilité induit une surcharge dans la communication, car à chaque fois le « Cluster Head » envoie un renifleur à chaque nœud du cluster et le (s) correcteur (s) visite (nt) le (s) nœud (s) défectueux. La surcharge dépend de la période de la collecte des paramètres (T) et du taux de pannes dans le réseau. T devrait être aussi large que possible pour réduire la surcharge de communication, mais en même temps juste assez petit pour assurer la fiabilité du réseau.
- l'imitation de la nature par le langage « danse des abeilles » le plus efficace pour conserver l'énergie des abeilles. Celles-ci sont très petites et ont des ressources limitées en énergie et en taille (le cas des capteurs). Les abeilles (agents mobiles) ne dansent qu'en cas de besoin (nécessaire) et leur danse n'implique qu'un nombre limité suffisant d'abeilles. L'algorithme essaie de profiter de cela pour les capteurs en diffusant des messages courts, limitant le nombre des messages diffusés le plus possible et en traitant les pannes d'une manière hiérarchique.

H. Outil distribué, agent mobile, à base d'arbre de décision

TinyD2[74] est une approche *réactive* de diagnostic des fautes. Contrairement, aux approches proactives, qui collectent les informations de diagnostic périodiquement des nœuds capteurs, TinyD2 adopte une approche *réactive* où l'échange d'informations ne commence qu'à la suite d'une détection d'un symptôme anormal. Selon le type de symptôme observé, la décision finale de diagnostic peut être prise localement en fonction des preuves observées sur le nœud local (par exemple, le redémarrage du système, peut être détecté et identifié par le nœud lui même), ou globalement en fonction des informations sur d'autres nœuds du réseau (par exemple, si un nœud détecte la panne franche d'un nœud, il faut vérifier le résultat avec les nœuds voisins). Dans ce dernier cas, le diagnostic se fait d'une

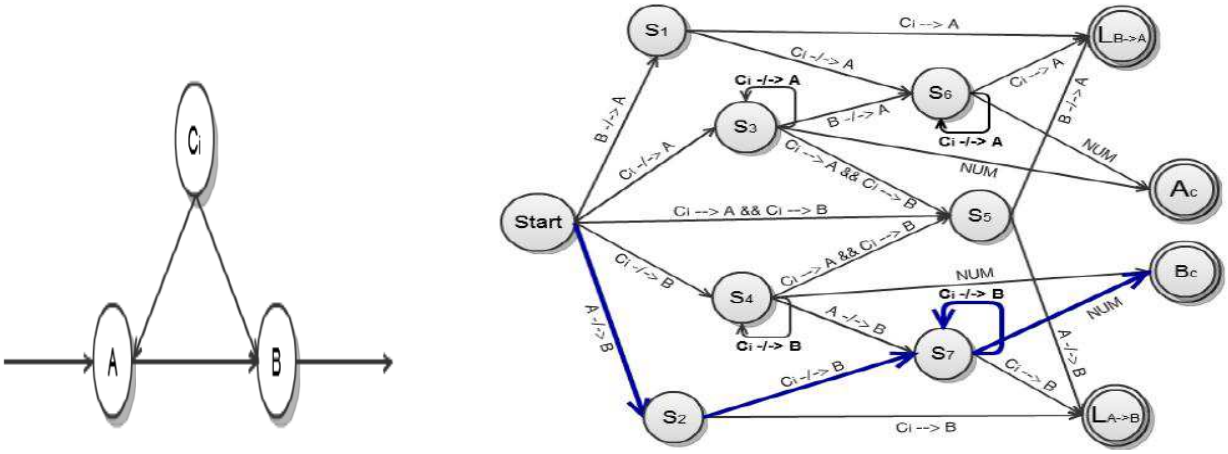


Figure 2-14: Un exemple du détecteur de défaut [74]

manière incrémentale ou graduelle où chaque nœud effectue une étape intermédiaire de décision jusqu'à l'arrivée au jugement final. Pour cela, chaque nœud maintient un ensemble de détecteurs de fautes dont chacun est représenté par une machine à états finis (Figure 2-14). Ainsi, chaque nœud participe dans le processus de diagnostic en faisant transiter la machine d'un état à un autre sur la base d'informations locales. Le détecteur, un message de petite taille incluant *principalement* le type de détecteur et son état courant, se déplace continuellement d'un nœud à un autre pour collecter plus d'informations. Dès qu'un nœud arrive à déduire la nature de la faute produite, il informe le puits pendant la phase de la collecte des notifications.

Voici un exemple montrant la coopération entre les nœuds A, B et {Ci}, un groupe de nœuds voisins de A et B, pour identifier la cause d'une retransmission élevée des paquets détectée sur le lien entre A et B. Le détecteur de la retransmission, détenu par chaque nœud du réseau, est encodé par une machine à états finis. L'état initial (S) représente l'état du nœud A qui crée en premier le détecteur. Les états finaux ($L_{B->A}$), (A_c), (B_c) et ($L_{A->B}$) représentent respectivement les causes potentielles du problème détecté : la mauvaise qualité du lien de B vers A, la congestion en A, la congestion en B et la mauvaise qualité du lien de A vers B. Pour passer à l'état suivant (S1, S2, S3 ou S4), le nœud A diffuse un message contenant le type et l'état courant du détecteur aux nœuds voisins. Chacun de ces derniers analyse le message et effectue des transitions en fonction des informations locales de chacun d'eux. Dans ce cas, le nœud B, qui a reçu un nombre des paquets inférieur à celui qui a été prévu au nœud A, fait transiter le détecteur à l'état S2 (signifie qu'il y a un problème de lien de A vers B). Il diffuse un message incluant l'état courant du détecteur S2. Si les nœuds Ci ont réussi à envoyer des paquets au nœud B, on passera à l'état final ($L_{A->B}$), sinon on déduit qu'il y a une contention en B, l'état (B_c).

I. Système distribué, à base d'échange de messages (protocole)

Dans [85], les auteurs présentent une solution de tolérance aux fautes au niveau des « clusters Head ». Une passerelle est considérée défectueuse

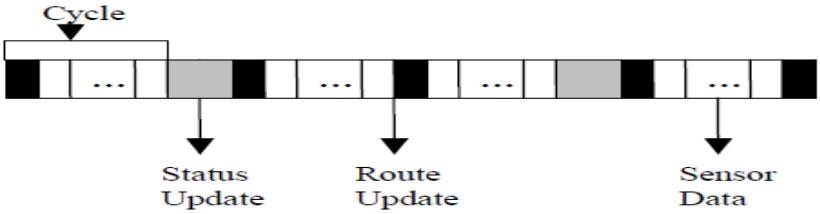
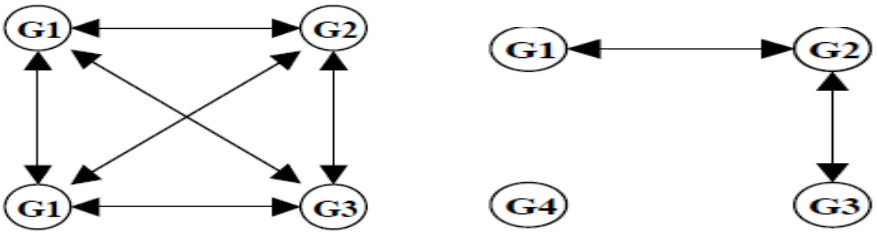


Figure 2-15: Allocation d'un créneau dans les réseaux de capteurs sans fils [85]

lorsqu'aucune autre passerelle du réseau n'arrive à communiquer avec elle. La détection de panne se base sur un échange des messages de statut entre passerelles et sur un mécanisme de consensus établi entre les passerelles qui sont en état correct. Le protocole de communication utilisé est le « TDMA MAC » qui alloue un créneau pour l'échange du statut de passerelle (Figure 2-15).

Un tableau de connectivité est crée suite à l'échange des messages de statut. Il représente les connexions entre les différentes passerelles et permet la détection de la défaillance d'une passerelle. Prenons les deux exemples suivants. Le Tableau2-3 représente la connectivité du réseau sans défaut de la Figure 2-16 où chaque passerelle peut communiquer avec toutes les autres passerelles ; et le Tableau 2-4 représente le réseau avec défaillances de la Figure 2-16 (b) dans lequel G4 est défectueuse et le lien entre G1 et G4 est rompu. En examinant le Tableau 2-4, les passerelles G1, G3 et G2 détectent que G4 est défectueuse (pas de communication avec toutes les autres passerelles).



(a) : Passerelles sans défaillances(b) Passerelles avec défaillances

Figure 2-16 : Pannes au niveau d'un « cluster head »

	G1	G2	G3	G4
G1	√	1	1	1
G2	1	√	1	1
G3	1	1	√	1
G4	1	1	1	√

Tableau2-3: Connectivité de l'exemple (a)

	G1	G2	G3	G4
G1	√	1	0	0
G2	1	√	1	0
G3	0	1	√	0
G4	0	0	0	0

Tableau 2-4: Connectivité de l'exemple (b)

Si G4 est défectueuse, les capteurs qui appartiennent à son groupe seront incapables de transmettre leurs données au puits. Le recouvrement consiste à associer ces capteurs à une autre passerelle qui est en bon état. L'approche proposée évite le mécanisme de « Re-clustering » à cause de son coût élevé. Par contre, c'est lors de la phase de l'initialisation du réseau (la phase de formation des clusters) que chaque passerelle maintient une liste des capteurs qu'on peut réassocier en cas de panne dans son groupe.

J. Outil distribué, actif, à base d'échange des messages

Memento [86] est un système de détection des défaillances. Les nœuds envoient périodiquement des messages de type « bitmap » à leur nœud parent. Un bitmap de forme 1101110 et de type « alive » signifie que les nœuds 3 et 7 sont défaillants. Les bitmaps de chaque nœud représentent les états des nœuds descendants. Memento

incorpore un protocole de communication qui permet de réduire la consommation d'énergie à 80% en comparaison aux méthodes standards de la collecte des données. Ce protocole inclut deux mécanismes :

- Mécanisme d'agrégation : les bitmaps des nœuds descendants sont agrégés au niveau de chaque nœud parent en appliquant l'opérateur binaire « ou ».
- Mécanisme de cache : un nœud parent cache l'état des nœuds descendants. Ces derniers s'abstiennent de transmission s'il n'y a pas de changement des états des nœuds.

Memento n'utilise pas de mécanisme d'acquittement. Par contre, les nœuds envoient d'une manière proactive un certain nombre de messages à chaque intervalle de temps, alors que ce nombre est estimé en fonction de la performance de la qualité de lien et du taux de faux positifs ciblé. Ce dernier est un paramètre de l'algorithme de détection. Memento est indépendant du type de l'application. En outre, il tolère la perte des messages dans la communication multi-saut.

2.2.4. Test et débogage d'applications

Les tests sont la première mesure pour pallier les bugs. Le débogage est l'activité qui consiste à diagnostiquer et corriger des bugs. La plupart des approches de test d'applications et de systèmes de RCSF sont appliquées hors ligne en raison de la forte consommation des ressources souvent nécessaire à la vérification et au débogage d'applications. Le test hors ligne est réalisé avant le déploiement du système dans des environnements contrôlables, ou après le déploiement à l'aide de programmes qui s'exécutent indépendamment de l'activité du système. Lors du débogage en ligne, le débogueur exécute le logiciel pas à pas et effectue après chaque pas une série de vérifications. Lors du débogage post-mortem, le débogueur examine un logiciel à la suite d'un crash du logiciel.

2.2.4.1. Origines (ou causes) des fautes logicielles

Parmi les causes des fautes logicielles, nous citons :

- Il est difficile de prédire tous les événements aléatoires ou non-déterministes [87][88] (la duplication des paquets, la perte des paquets, les pannes des nœuds et des liens, le redémarrage des nœuds, la corruption des paquets, l'interaction complexe entre les nœuds, *etc*.) pouvant survenir pendant l'exécution du système. En effet, ces événements paralysent le fonctionnement de l'application [89][90][91][92] en révélant des « bugs » qui n'ont pas été détectés pendant les phases de validation dans un environnement contrôlé (validation avant le déploiement). De plus, une étude de terrain des déploiements de RCSF montre que les « bugs » qui se manifestent dans les applications de RCSF sont souvent de nature transitoire, occasionnelle et non reproductible [79]. D'où, la préoccupation des testeurs qui se focalisent fortement sur la modélisation des phénomènes aléatoires. Le but est d'explorer le plus grand nombre de chemins d'exécution possibles et

éventuels pour détecter efficacement « des bugs » résultant de l'intégration et de l'interaction du système avec un environnement dynamique et aléatoire.

- Compte tenu des contraintes de ressources des capteurs, un système d'exploitation basé sur des événements et modèle concurrentiel de programmation, peut générer des problèmes de concurrence (deux processus qui veulent accéder a une même donnée) « race conditions », de dépassement de la pile « stack overflow » et des interblocages « deadlock ».

2.2.4.2. Approches avant le déploiement

Avant le déploiement, plusieurs outils peuvent être utilisés pour détecter les bugs logiciels, valider la conformité de l'application en fonction des besoins exprimés, et vérifier le processus de développement, parmi lesquels :

- Les simulateurs offrent un environnement qui teste les protocoles d'une manière bénéfique en termes de temps et de coût. En effet, pour de grands réseaux le nombre de capteurs peut atteindre plusieurs milliers et donc un coût financier relativement important. Une phase de validation par simulation serait très bénéfique. Elle permet de réduire au maximum les erreurs de conception possibles, de réaliser des scenarios diversifiés et complexes de déploiement dans un temps raisonnable et d'économiser les coûts de mise en œuvre. Par contre, les simulateurs ne peuvent pas modéliser avec une bonne précision toutes les conditions de déploiements réels. Parmi les simulateurs connus : TOSSIM[93], COOJA[94], OMNET++[95], NS 2[96],GloMoSim[97], WSNet[98], J-Sim[99], *etc*.

- *Les outils de débogage* ont pour objectif de détecter les bugs logiciels. Kleenet [100]détecte les bugs résultant de l'interaction complexe entre les nœuds, les événements non déterminismes et les entrées imprévues. Kleenet assure une haute couverture de code en testant l'application avec des entrées symboliques et en injectant des événements aléatoires. T-check [101] détecte les bugs par des techniques de marches aléatoires et un modèle de vérification à états explicites. Dans [102], l'outil proposé capte l'entrelacement des séquences d'opérations de l'application par des techniques avancées de modélisation les flux de contrôle.

Les difficultés de maintenance et d'accès après le déploiement rendent les approches appliquées avant le déploiement indispensables pour assurer la fiabilité des applications. Ces approches permettent d'analyser le comportement des applications dans des conditions proches de la réalité. Toutefois, la vérification des applications avant le déploiement ne suffit pas pour éviter ou prédire des fautes qui ne sont observables que lors de l'intégration du système dans un environnement hostile, non contrôlable et imprévisible. Il est donc nécessaire d'intégrer des outils de vérification qui visent à détecter les défaillances de l'application en temps réel lorsque le système est en fonctionnement.

2.2.4.3. Approches après le déploiement

Les approches de test et de débogage après le déploiement sont nombreuses, parmi lesquelles : le débogage post-mortem, le débogage à distance, le test par assertion et le test fonctionnel.

A. Débogage post-mortem

Le débogage post-mortem permet le débogage d'une application déployée sur un RCSF. Il considère des plateformes variées. Le débogage post-mortem s'effectue en deux étapes :

- La journalisation des traces d'exécution ou des messages de communication.
- L'analyse des traces par des techniques d'apprentissage.

Dustminer [79] se focalise sur les bugs dûs à une interaction inappropriée et imprévue des composants éventuellement non défaillants du système. Ce type de bugs est difficile à détecter parce qu'il est de nature transitoire, occasionnelle et non reproductible. Dustminer vise à détecter la séquence d'événements (bugs liés à plusieurs composants au contraire des bugs de type : mauvais pointeur) qui provoque une dégradation considérable des performances d'un protocole. Pour cela, il applique l'algorithme Apriori[1] pour classifier les conditions liées au comportement normal et celui anormal du système. Pendant la phase d'apprentissage, Dustminer étiquète les séquences d'événements en deux classes en se basant sur des prédicats définis par le développeur de l'application. Dustminer a été efficace dans l'identification des causes principales de la dégradation des performances d'un nouveau protocole MAC conçu pour la plateforme TinyOS et a permis d'identifier des bugs dans le kernel du système LiteOS. Dustminer ne peut identifier les bugs au niveau du code source. De plus, Dustminer requiert une intervention du développeur pour étiqueter les journaux d'événements.

B. Débogage à distance

Le débogage à distance offre la possibilité de contrôler l'exécution de l'application, en permettant par divers moyens d'arrêter l'exécution de l'application et d'observer par exemple le contenu des différentes variables en mémoire. L'état d'exécution peut alors être observé afin, par exemple, de déterminer la cause d'une défaillance.

[1]L'algorithme Apriori est un algorithme d'exploration de données, dans le domaine de l'apprentissage des règles d'association. Il sert à reconnaître des propriétés qui reviennent fréquemment dans un ensemble de données et d'en déduire une catégorisation.

– **Débogage au niveau du code source**

Clairvoyant[61] : débugue l'application au niveau du code source sans nécessiter la modification du code de l'application (instrumentation dynamique binaire). Il supporte les commandes de débogage traditionnel, par exemple « break », « step », et des commandes réseaux (commandes à un groupe de nœuds), par exemple commandes d'appel des interruptions, synchronisation des nœuds, *etc*. Le système est formé d'un composant résidant sur chaque nœud (l'interpréteur de commandes) et un composant résidant sur le nœud central à partir duquel le développeur envoie les commandes de débogage via un terminal de type « GDB ». L'espace mémoire occupé par le composant du nœud est égal à 1 K-octets RAM et 32 K-octets de flash. Clairvoyant a pu identifier des dysfonctionnements de dépassement de pile et d'interblocage dans TinyOS 1.x.

Marionette [60] permet l**'**appel des fonctions, la lecture et l'écriture des variables des nœuds distants. Le développeur doit marquer les fonctions à appeler au niveau du code source de l'application avant le déploiement. Cet outil permet au développeur d'ajouter des fonctionnalités après le déploiement en écrivant des scripts en Python au niveau du nœud central. L'espace mémoire occupé par le composant du nœud est égal à 153 octets RAM et 4 K-octets ROM. Marionette ne supporte pas toutes les fonctionnalités de débogage « breakpoints », « watches » et « traces ».

– **Débogage au niveau système**

NodeMD[103] est un outil conçu pour le système d'exploitation multithread **«** Mantis **».** NodeMD instrumente le code du nœud pour identifier les dysfonctionnements de dépassement de la pile et d'interblocage. Il adopte l'approche « aspect-like » qui vérifie le dépassement de la pile à chaque appel d'une procédure et sans utiliser un nombre excessif de cycles. NodeMD associe un temporisateur pour chaque « thread » dont la valeur est égale au double de sa durée d'exécution. Cet outil détecte les interblocages du système à l'expiration du temporisateur

C. Tests par assertion

Les tests par assertion permettent au développeur de formuler et d'évaluer des

Comportement normal	Détection d'erreur
If faut un seul CH à tout moment	un nœud a reçu un message « keep alive », indiquant la présence d'un CH, de plusieurs CH différents (détection au niveau nœud).
L'élection de CH ne doit être faite qu'une fois à chaque période T	Un message d'initialisation de l'élection est généré plusieurs fois pendant la période T (détection au niveau Cluster).

Tableau 2-5: Invariants et méthodes de détection pour les protocoles de « clustering »

assertions pendant l'exécution de l'application. **PDA [104]** permet au développeur de formuler des assertions sur les états des nœuds (variable locale, attribut dans d'autres nœuds) en utilisant un langage déclaratif simple. PDA évalue les assertions pendant l'exécution de l'application. En cas d'évaluation échouée, il envoie des notifications au développeur en indiquant la localisation des assertions au niveau du code source. PDA est indépendant du type de l'application. Il supporte plusieurs stratégies de collecte d'informations des nœuds : canal de l'application, réseau de renifleurs, réseau filaire ou sans fils, banc de test. Le développeur peut choisir la stratégie en fonction d'un compromis entre le coût de mise en œuvre et la transparence attendue par l'outil. PDA fournit la fonctionnalité de synchronisation les traces indépendamment de la synchronisation des nœuds. En outre, PDA prévoit des mécanismes pour faire face aux traces erronées qui résultent de la perte de messages. Cependant, PDA ne permet pas l'ajout des assertions après le déploiement de l'application.

HSEND [105] (Hierarchal Sensor network Debugging) est une approche semi automatique pour la détection des pannes. Dans cette approche, le programmeur spécifie l'exactitude des propriétés du protocole appelées « invariants ». Ceux-ci sont associés aux conditions (les variables observées) que ce soit au niveau du nœud ou au niveau du réseau. Le compilateur insère automatiquement les codes de vérification pour assurer que les variables observées satisfont les invariants. La panne est détectée à la violation de ces invariants (Tableau 2-5). Dans ce cas, le programmeur procède au transfert des nouveaux programmes ou des patches afin de traiter la panne.
La détection peut être locale ou globale. Elle est locale lorsqu'elle dépend des invariants locaux au nœud. Elle est globale si elle dépend des variables et conditions associées à plusieurs nœuds du réseau. La vérification des variables globales engendre une surcharge d'énergie pour communiquer les informations de débogage au cluster Head. Afin d'économiser la consommation d'énergie, l'approche H-SEND adopte le mécanisme de marquage où les variables requises au débogage sont superposées avec les données du réseau. De plus, la simulation a montré que la détection hiérarchique d'erreurs (nœud-CH-puits) a réduit la surcharge de 7% par rapport à l'approche centralisée qui consiste à envoyer les informations de débogage au puits.

D. Test en ligne

Le but du test est de vérifier si l'application fonctionne correctement en dépit de la défaillance d'un ou plusieurs éléments du système, de la topologie dynamique, de la reconfiguration ou la réinitialisation des nœuds, de l'occurrence d'attaques, *etc*.

RTA [106] est une plateforme pour la conception et le développement d'applications intégrant le test en ligne **(**Figure 2-17**)**. RTA adopte un langage de spécification de haut niveau appelé « SNEDL »[107]. Le langage permet de représenter les propriétés structurelles, temporelles et spatiales de n'importe quelle application à base de détection d'événements. Ce langage a trois avantages principaux :

– La possibilité de représenter les applications de logique complexe et les états d'incohérence par des modèles simples.

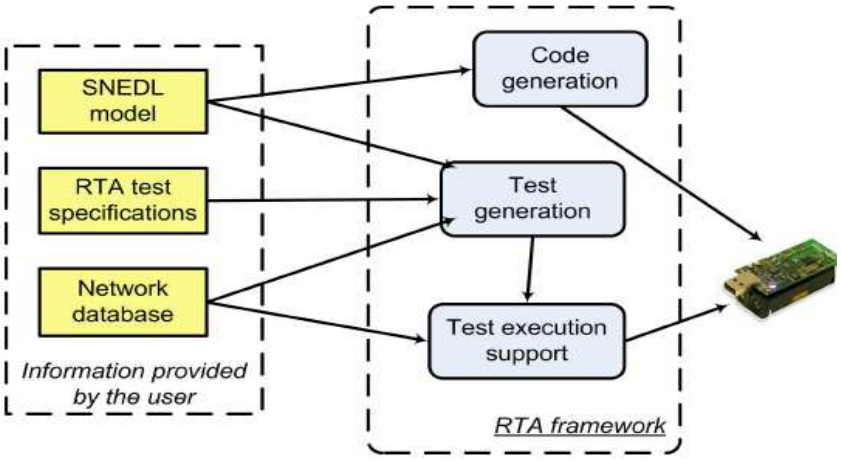


Figure 2-17: La plateforme RTA [106]

– Le modèle de flux de jetons permet de faire des tests en ligne. Il suffit de marquer les jetons de test par un identifiant différent de celui des jetons de l'application réelle. Le langage facilite la journalisation des traces d'événements pendant le fonctionnement du système.

Le développeur doit spécifier le modèle d'application, les tests et la topologie du réseau. La spécification des tests inclut l'ensemble des associations entrées-sorties, la date de lancement des tests, les nœuds concernés par le test, *etc*. Les codes des applications et des tests sont générés par un générateur de code « TinyOS ». Ce dernier a la capacité de partitionner le modèle pour les différents types de nœuds « clusterhead », nœuds capteurs. Le générateur peut être adapté pour générer le code en d'autres langages et pour d'autres plateformes. Au déclenchement du test, chaque nœud concerné par le test lit les mesures virtuelles, exécute toutes les opérations nécessaires (le calcul, la communication des messages, les opérations distribuées, *etc*.) et produit en sortie des événements virtuels. Les sorties du test sont ensuite comparées à celles prévues (pour valider le comportement du système. Les cas de test doivent être déterminés de façon qu'ils assurent la couverture totale des chemins d'exécution du programme. Mais, en même temps, ils doivent être réduits le plus possible pour réduire l'utilisation des ressources limitées des nœuds capteurs. Dans ce but, des techniques de réduction des cas de test ont été introduites en exploitant principalement la topologie du réseau et la redondance des nœuds. Concernant le premier facteur, un réseau peut être divisé en plusieurs unités logiques ou régions dont chacune est formée d'un certain nombre de nœuds voisins réalisant une activité commune. Il est possible de tester chaque région séparément des autres. La redondance permet d'utiliser les mêmes valeurs pour les nœuds redondants. L'étude expérimentale a montré que RTA est économe en énergie en le comparant à l'outil de surveillance Memento [86]. RTA induit 33% de messages circulant en moins dans le réseau que Memento.

2.2.5. Algorithmes de conservation d'énergie

L'ensemble des algorithmes (à l'exception de « data checkpointing »), que nous avons présentés ci-dessus, sont de type *curatif*, puisque le mécanisme de tolérance aux pannes implémenté n'est exécuté qu'après la détection des pannes. Par la suite, nous présentons une autre famille d'algorithmes qui sont *préventifs*, i.e. qui tentent de retarder ou éviter tout type d'erreur afin de garder le réseau fonctionnel le plus longtemps possible. La conservation d'énergie, à titre d'exemple, permet de consommer moins d'énergie et évite donc une extinction prématurée de la batterie ce qui augmente la durée de vie des capteurs. Plusieurs solutions sont mises en place : l'agrégation des données, la compression des données, la fusion des données, le « clustering » et la mobilité**.**

2.2.5.1. Agrégation des données

L'agrégation permet d'économiser l'énergie pendant la collecte des données en limitant le nombre de messages nécessaires à leur remontée vers la passerelle d'accès. Cette approche est décrite dans la plupart des propositions récentes du domaine. Il s'agit de profiter du nature multi-saut des protocoles de routage. Chaque capteur agrège sa mesure avec celle de son prédécesseur (par exemple il l'additionne) avant de transmettre la réponse au capteur suivant. Cette agrégation peut soit utiliser un opérateur classique (somme, moyenne, minimum,..) sur les mesures des différents capteurs soit effectuer un simple ajout d'une mesure à une autre dans un même paquet réseau. Ainsi, le nombre de messages envoyés est minimisé et l'énergie consommée est moindre comparée à une solution où tous les capteurs communiquent directement avec la passerelle. La référence [108] présente un algorithme d'agrégation et de consensus des données, conçu pour les applications de localisation des objets dans un réseau de capteurs sans fils. L'objectif du consensus est de réduire le nombre des messages envoyés des sources vers le puits. Pour cela, les nœuds qui se trouvent dans une même zone de couverture et ont un intérêt commun, forment des groupes. Ils échangent des informations entre eux pour arriver à une décision finale (un consensus). Celle-ci est la seule à être envoyée au puits. Le consensus permet alors une réduction dans la consommation d'énergie et un prolongement de la durée de vie du réseau. Le consensus repose sur un mécanisme de quorum qui le rend efficace et robuste en présence de pannes dans le réseau.

DADMA[109] est un algorithme d'agrégation distribuées ou les capteurs procèdent à l'agrégation/dilution des données selon des règles spécifiées dans des requêtes SQL. L'étude expérimentale a montré que DADMA permet de réduire le nombre de paquets transmis de 60%.

FEEDA [110] est un mécanisme fiable d'agrégation des données. Il permet la conservation d'énergie en limitant le nombre des réponses redondantes envoyées des nœuds capteurs. Il est tolérant aux pannes grâce au mécanisme des chemins redondants établis dans une structure arborescente où chaque nœud de l'arbre

maintient deux parents : primaire et secondaire. Le père secondaire remplace le primaire en cas de panne. Les parents se coordonnent pour agréger la valeur une seule fois. L'algorithme FEEDA a introduit un drapeau « Fault status » à chaque paquet transmis indiquant si le nœud parent a bien reçu la réponse éventuelle de chaque nœud fils. Si un nœud envoie un message au nœud primaire, celui-ci envoie la valeur agrégée en mettant le drapeau à zéro. Sinon, il envoie sa valeur locale et met le drapeau à 1 et le père secondaire, qui espionne les liens qui le relient au père primaire, se charge d'envoyer la valeur agrégée.
Dans **Cougar [111],** les données produites par le réseau de capteurs sont modélisées comme une table relationnelle. Dans cette table, chacun des attributs représente soit des informations sur le capteur ou bien des données produites par ce capteur. L'approche Cougar fournit une agrégation partielle au niveau des capteurs. Chaque capteur maintient une liste d'attente contenant les capteurs fils qui doivent lui envoyer les paquets. Le capteur n'émet le paquet agrégé au prochain saut que s'il a reçu les paquets de tous les capteurs de la liste d'attente. Cependant, un capteur peut devenir inaccessible à cause du mouvement ou d'un problème de batterie. Pour cela, Cougar utilise un temporisateur afin d'éviter une attente indéfinie. Dans le but de conserver l'énergie, une technique de fusion des données est présentée dans[112]. Elle consiste à classifier les données captées en deux catégories : les données émergentes et les données usuelles. Les données émergentes sont celles qui changent fréquemment comme les vibrations des volcans. Par contre, les données usuelles changent rarement comme la température dans une chambre. Pour minimiser la redondance inutile, les données transmises au puits sont seulement les données émergentes et le changement dans les données usuelles. Pour cela, l'algorithme introduit une couche de fusion des données entre la couche application et la couche de routage pour effectuer la fusion.

2.2.5.2. Clustering

Les algorithmes proposés dans la catégorie du « clustering » viennent au secours du problème d'auto-organisation du réseau. Les protocoles de « clustering » divisent le réseau en un ensemble de clusters ayant chacun un capteur hôte (Cluster Head) qui récupère les données depuis tous les capteurs de son cluster puis les achemine vers le puits. Cette solution permet de mieux gérer le trafic du réseau et d'alléger la quantité d'informations qui circulent, en effectuant des traitements au sein du cluster avant de propager les données vers le reste du réseau pour les transmettre au collecteur. Dans [113], les auteurs proposent un protocole de « clustering » tolérant aux pannes avec une gestion efficace de l'énergie. Ce protocole tolère les pertes des messages grâce aux mécanismes d'acquittement/retransmission et des routes redondantes créées au niveau inter cluster à la phase d'initialisation. Il gère efficacement l'énergie grâce au choix de chemins alternatifs, à la dissémination efficace des données à travers les « clusters Head » et les voisins les plus proches entre les « clusters », la possibilité d'entrer dans l'état de « sommeil » pour les nœuds inactifs et l'ordonnancement TDMA déclenché par le «ClusterHead ».

2.2.5.3. Mobilité

Certains protocoles proposent comme solution tolérante aux pannes la sélection d'un ensemble de nœuds mobiles chargés de se déplacer entre les capteurs pour collecter les données captées. Ceci réduira l'énergie consommée au niveau de chaque capteur en éliminant sa tâche de transmission. Un nœud mobile est généralement doté d'une batterie plus importante que celle d'un nœud capteur. Une solution de tolérance aux pannes se basant sur la **mobilité**, est présentée dans [83]**.**

2.3. Comparaison des approches

Tenant compte des problématiques traitées dans cette thèse ainsi que de l'importance d'étudier l'impact de certains critères sur la validation d'une solution globale de détection des fautes adaptée aux RCSF, nous avons comparé les approches présentées selon six critères : la couverture des fautes et la nature des fautes, le passage à l'échelle, l'hétérogénéité, la flexibilité, la transparence et la robustesse (Tableau 2-6). Dans ce tableau, nous désignons par le nombre des signes + et – le degré d'impact des critères de chaque approche présentée.

La couverture des fautes : nous désignons par ce critère le domaine (nœud, réseau, application, données, global) détectable par l'outil d'inspection.

- Les approches centralisées et actives ont une couverture la plus élargie des fautes. Les approches centralisées se basent sur des informations recueillies de l'ensemble des nœuds du réseau. De ce fait, elles peuvent identifier des critères globaux de performance (fiabilité, coût, latence, *etc*.). En outre, l'approche active permet l'accès à l'état interne du nœud. Mais, elles ne peuvent pas identifier les fautes au niveau de la sémantique de l'application.
- Les approches à base de marquage sont similaires aux approches actives mais le nombre des métriques qui peuvent être transmises par un paquet est limité (le paquet qui circule a une taille maximale)
- Les approches passives à base de renifleurs ne permettent pas l'accès à l'état interne des nœuds.
- Les approches distribuées à base de protocoles ne peuvent pas identifier les problèmes globaux (latence, fiabilité, *etc*.).
- Les approches de débogage à distance sont limitées aux fautes au niveau des nœuds.
- Les tests par insertion de prédicats sont spécifiques aux fautes de conception des protocoles et des applications.
- Le test d'application ne peut pas diagnostiquer les fautes au niveau des nœuds et du réseau ni les fautes au niveau des données.

La nature des fautes : nous désignons par ce critère la capacité de l'outil à adresser des fautes prédéfinies ou fautes silencieuses (détection agnostique), d'une manière statique (fixe) ou dynamique (extensible).

- Les approches à base d'arbre de décision traitent des fautes prédéfinies de manière statique.
- Les approches à base d'inférence probabiliste traitent des fautes prédéfinies de manière extensible.
- Les approches à base de corrélation spatiale et temporelle des métriques traitent des fautes silencieuses au niveau nœud et réseau.
- Les approches à base d'apprentissage et à base de test traitent souvent des fautes agnostiques au niveau nœud, réseau et application.
- Les approches à base de débogage (inspection des variables) traitent des fautes de manière extensible.

Approche	Couverture des fautes	Passage à l'échelle	Hétérogénéité	Flexibilité	Transparence	Robustesse
Diagnostic à base de renifleurs	+	+++	-	-	++++	+++
Diagnostic à base de marquage des paquets	++	+++	-	+	+++	++
Diagnostic actif centralisé	+++	+	+	+	+	+
Diagnostic actif distribué	++	+++	+	---	++	+++
Diagnostic à base d'agents mobiles	++(+)	++(+)	+	+	+++	+++
Diagnostic post mortem	++	++	+	---	++	++
Débogage à distance	+	+	+	+	+	++
Test par assertion	++	++	+	----	++	++
Test d'application	++	+++	+	+	+++	++

Tableau 2-6 : Comparaison des approches

Passage l'échelle : Le passage à l'échelle est la capacité d'augmenter la taille du réseau sans avoir besoin d'augmenter les ressources proportionnellement à la taille du réseau, ni d'augmenter l'effort humain à traiter les sorties des outils d'inspection.

- Les approches actives et centralisées et les approches à base de débogage à distance ne permettent pas le passage à l'échelle. Les approches actives et centralisées génèrent des paquets de contrôle à transmettre sur une communication multi-saut. D'une part, ces paquets affectent la performance et la fiabilité du système et d'autre part ils consomment la bande passante du réseau. Les approches à base de débogage requièrent un effort humain pour interagir avec chaque nœud individuellement.
- Les approches de test par insertion sont des approches semi automatiques. Ces approches favorisent les réseaux de taille moyenne.
- Les approches passives, les approches post-mortem, les approches distribuées à base de protocoles et à base d'agents mobiles permettent le passage à l'échelle.

Hétérogénéité : l'hétérogénéité est la possibilité de supporter des nœuds d'architecture matérielle et logicielle différente. Dans notre cas, il s'agit de supporter des applications différentes et des capteurs différents.

- Les approches passives (à base de renifleurs, à base de marquage des paquets) ne conviennent pas aux applications à base d'événements. Ces approches se basent sur le flux des paquets pour récupérer les informations de diagnostic.
- Les approches post mortem supportent des plateformes variées.
- Les approches actives, de débogage à distance et les tests sont indépendantes de l'application.

Adaptabilité : l'adaptabilité correspond au support des changements environnementaux, en particulier la mobilité, l'ajout et le retrait des nœuds.

- Les approches à base de renifleurs supportent la mobilité dans la limite de la portée de la communication radios des renifleurs.
- En général, les approches qui utilisent la même pile protocolaire que l'application supportent la mobilité.

Transparence : nous désignons par transparence, la capacité de surveiller un système en évitant ou en minimisant tout impact négatif sur les performances de ce système.

- Les approches de diagnostic actif (en particulier celles proactives), du débogage et du test génèrent des paquets de contrôle qui augmentent le risque d'interférence/congestion/*etc*. et affectent la livraison des paquets de l'application réelle. En outre, l'incorporation du code pour envoyer les paquets de contrôle affecte la fiabilité du nœud.
- Les approches passives à base de renifleurs sont tout à fait transparentes.
- Les approches à base de marquage et à base d'agents mobiles ont un impact minimal sur l'activité du système.

Robustesse : la robustesse détermine la capacité à surveiller un système indépendamment des fautes de tel système.
Les approches centralisées ne sont pas robustes. La panne du nœud central affecte le système en entier. De plus, elles sont sensibles aux fautes de communication multi-saut (perte des messages).

- Les approches distribuées et les approches qui utilisent un réseau dédié, généralement plus fiable que le RCSF sont robustes vis-à-vis aux fautes du système.
- Les approches à base d'inférence tolèrent dans une certaine limite les fautes de communication (la perte des messages).

2.4.Positionnement de notre travail

Dans ce chapitre, nous avons présenté des solutions d'inspection des fautes à différents niveaux (données, nœud, réseau et application) ainsi que des solutions de conservation d'énergie permettant d'augmenter la durée de vie des capteurs. L'étude de ces différentes solutions nous mène à tirer les remarques suivantes :

- Les outils existants inspectent généralement une liste prédéterminée et limitée de fautes. Par exemple, les outils à base de renifleurs ne permettent pas l'accès à l'état interne des nœuds, les outils à base d'arbre de décision ne peuvent représenter toutes les situations (y compris les défauts multiples), *etc*. Donc, une solution globale de détection des fautes n'existe pas. Une telle solution devrait traiter les fautes à différents niveaux (données, nœud, réseau et service d'application).
- Le passage à l'échelle est traité différemment par les outils d'inspection existants. Par exemple, les approches centralisées et celles de débogage à distance ne favorisent que les réseaux à petite ou à moyenne échelle ; les réseaux de renifleurs favorisent le passage à l'échelle mais nécessitent un coût d'installation supplémentaire ; les approches distribuées favorisent les réseaux à grande échelle en introduisant des mécanismes permettant de contrôler le nombre des paquets de contrôle circulant dans le réseau, *etc*. Ici, en présence de plusieurs types d'outils de détection de fautes fonctionnant en ligne, il sera indispensable d'introduire plusieurs types de mécanismes de gestion de ressources pour éviter tout impact sur la performance du système à base de RCSF.
- Les outils existants sont souvent spécifiques à un type de plateforme, à un type d'application ou un type de capteur. Par exemple, les réseaux de renifleurs et les outils à base de marquage sont spécifiques aux applications de la collecte des données. De nombreux des outils sont conçus uniquement pour la plateforme «TinyOS », *etc*. En effet, les problèmes de portabilité au niveau plateforme sont des problèmes ouverts dans le domaine de RCSF.
- La conception d'un outil d'inspection robuste sans avoir besoin d'installer des matériels supplémentaires est toujours un objectif difficile à réaliser. En effet,

le système de diagnostic utilise la même infrastructure (les nœuds, les liens, les protocoles de communication, *etc*.) que celle du système surveillé qui à son tour se caractérise par un taux des fautes élevé.

Par rapport à cet existant, le positionnement de notre travail de thèse se fait sur les trois points suivants : la couverture des fautes, le passage à l'échelle et l'hétérogénéité. Les deux premiers points sont liés l'un à l'autre. En effet, une couverture élargie des fautes permet un fonctionnement correct des systèmes de RCSF. Cependant, un tel objectif impose une politique rationalisée dans l'utilisation des ressources pour maximiser la durée de vie du réseau d'une part et permettre le passage à l'échelle d'autre part. Le dernier point consiste à concevoir une solution le plus possible générale pour être exploitée dans de nombreux contextes de déploiement.

- *La couverture des fautes :* Le test d'application détecte les problèmes potentiels qui affectent le comportement des applications après le déploiement. Le test ne permet pas de dévoiler les causes des problèmes. Les approches de diagnostic adressent les fautes au niveau données, nœud ou réseau. Notre approche consiste à combiner les approches de test et de diagnostic en ligne pour assurer une couverture élargie des fautes. Nous présentons les trois parties constituant cette approche dans les chapitres 3, 4 et 5. Dans le chapitre 3, nous présenterons les fautes au niveau des données issues des capteurs. Nous proposons une nouvelle approche de détection et de classification des fautes fondée sur deux méthodes complémentaires de traitement de signal : corrélation temporelle et réseaux des neurones à base de corrélation spatiale. Dans le chapitre 4, nous présentons une approche générale et efficace de diagnostic des fautes au niveau nœud et réseau. L'approche diagnostique les causes de la réduction de débit des données (la défaillance d'un nœud due à l'épuisement de l'énergie et la défaillance des liens due à une mauvaise connectivité). Dans le chapitre 5, nous présentons le test d'application ainsi que l'approche globale d'inspection des fautes. Le test d'application vérifie si les différentes fonctionnalités offertes par l'application fonctionnent comme prévu.

- *Le passage à l'échelle* : la gestion d'énergie est réalisée par les mécanismes de contrôle le nombre de paquets de contrôle (chapitre 4), la gestion distribuée des fautes au niveau des nœuds et des «clusters Head» (chapitre 3 et 5) et le mécanisme d'isolation des fautes au niveau géographique et/ou au niveau des services d'application. Ce dernier mécanisme s'appuie sur une relation de dépendance entre les différents types de test pour les déclencher efficacement.

- *L'hétérogénéité* : notre solution est générique au niveau application et au niveau des capteurs. Nous bénéficions de l'approche orientée-service pour s'adapter aux différents types d'applications et différents types de capteurs.

2.5.Conclusion

La flexibilité, le coût réduit, et les moyens rapides de déploiement élargissent le champ d'applications des réseaux de capteurs sans fils. Cependant, la mise en place d'une application de réseau de capteurs doit prendre en considération les contraintes qui caractérisent les nœuds capteurs et qui sont principalement : la consommation d'énergie, et l'adaptation à un environnement hostile. Ces contraintes exigent que des solutions de gestion des fautes soient mises au point pour offrir des garanties de qualité de services aux applications. Dans ce chapitre, nous avons exploré des solutions existantes dans la littérature concernant les méthodes de détection et recouvrement des différents types de fautes. Par la suite, nous présenterons des solutions de gestion des fautes au niveau des données, nœuds, réseau et application proposées en vue de concevoir un framework global assurant une couverture élargie des fautes tout en maîtrisant la consommation d'énergie.

Chapitre 3.

FIABILITE DES DONNEES CAPTEURS

Une prise de décision basée sur des données erronées issues de capteurs peut entraîner des actions incorrectes, nuisibles ou coûteuses. Ceci pose la question de la fiabilité des données issues de capteurs. Dans ce chapitre, nous présentons des approches de détection de fautes des données développées dans le cadre d'une approche globale de sûreté de fonctionnement dans le RCSF.

SOMMAIRE

3.1. Introduction

Organisés sous forme de réseau, les capteurs (ou nœuds) d'un RCSF, malgré la limitation de leurs ressources, ont pour mission de récolter des données issues de phénomènes physiques et les faire parvenir à une station de base. En effet, l'utilisation de capteurs notamment au sein du domaine environnemental (i.e., pour la surveillance des inondations, des avalanches, des volcans,..) permet une interprétation plus simple du monde réel. Cependant, les grandes quantités de données issues de ces capteurs, à fréquences et positions variables dans le temps, acquises dans des environnements hostiles avec une capacité d'énergie et de communication limitée, rendent les données imprécises et incertaines.

Les données erronées sont souvent nuisibles à des prises de décision adéquates, et nuisent aux objectifs des utilisateurs. Ces utilisateurs peuvent être des volcanologues, des urbanistes, des biologistes…, qui élaborent des plans et des stratégies basés sur l'analyse de ces données. Ils sont ainsi souvent exposés au risque de traiter des données non fiables, s'ils ne possèdent pas de moyens permettant d'évaluer ou d'assurer le niveau de qualité de l'information. Par conséquent, la prise de décisions basée sur des données de qualité médiocre, peut entraîner des actions incorrectes comme le déclenchement intempestif d'une alerte, des représentations cartographiques erronées (i.e. routes inexactes,…), une activation coûteuse et inutile des actionneurs et jusqu'a l'endommagement complet du système d'information à exploiter. Ceci pose la question de la fiabilité des données issues de capteurs.

Le but de notre travail est donc de relever le challenge de la fiabilité des données issues de capteurs, afin d'aider les utilisateurs (volcanologues, experts, gestionnaires de l'environnement, gestionnaires du patrimoine urbain, *etc*.) lors de la prise de décision surtout en situations critiques. Ainsi, pour concevoir un système de détection/correction des erreurs sur les données, nous avons étudié les spécificités des données issues des capteurs. Ensuite nous avons proposé des méthodes adaptées aux RCSF pour détecter et/ou classifier les fautes des données capteurs. Finalement, nous avons adopté la solution à intégrer dans l'architecture de notre solution globale pour la détection des fautes.

3.2. Spécificités de données capteurs

Différents types de données peuvent être fournis par les nœuds/capteurs en fonction du type de l'application. Les données acquises, appelées aussi données d'observation sont souvent *hétérogènes [8][114][115][116]*. Elles correspondent à des éléments comme l'humidité, la température, la pression, *etc*., et peuvent être représentées sous différents formats, de façon numérique, analogique, spatiale, temporelle, spatio-temporelle, multimédia (image, vidéo), *etc*.

Une observation peut être réalisée en utilisant des capteurs, pendant des périodes de temps pour lesquelles les observations sont demandées, sur un phénomène quelconque ou sur une région qui contient plusieurs capteurs d'intérêt ou des éléments qui sont l'objet d'une analyse. De ce fait, les données issues des capteurs possèdent des *propriétés*

spatiales et *temporelles[117][118][119]*. Les capteurs liés à une observation, peuvent être référencés de façon spatiale. Une donnée liée à un capteur dans l'espace, peut porter des estampilles de temps pour notifier la date de l'acquisition.

De plus, les données issues de capteurs peuvent être variables dans le temps, voire *dynamiques [8][120]*. En effet, ces données collectées dans des situations souvent difficiles, possèdent des caractéristiques qui évoluent au cours du temps.

Ces spécificités nécessitent donc un traitement particulier en termes de qualité prenant en considération l'hétérogénéité, les caractéristiques spatiales et temporelles et la dynamique éventuelle des données capteurs.

3.3.Positionnement

Les approches de détection des fautes de données présentées dans la littérature sont nombreuses. En générale, ces méthodes suivent une démarche commune : elles définissent un comportement normal des mesures de capteurs, puis considèrent les écarts significatifs du comportement normal comme des erreurs. Elles se différencient par les hypothèses considérées quant au phénomène observé.

- Une classe d'approches nécessitent un niveau de redondance dans le réseau pour détecter les fautes des données en comparant les mesures des nœuds voisins [51][52][53][54]. En effet, l'utilisation d'un grand nombre de capteurs et de plusieurs variétés donne plus de précision dans l'information recueillie par rapport à celle obtenue par un seul capteur. Cependant, ces approches ne classifient pas les types de fautes. La classification des types de fautes de données facilite la mise en œuvre d'un système de correction des données. En outre, ces approches peuvent être coûteuses en termes de communication des messages.

- Une autre classe d'approches nécessitent une connaissance, a priori du modèle de des données de capteurs [121][122]. Ces approches utilisent un modèle de prédiction des données sur la base de l'analyse des séries temporelles pour estimer les mesures de capteurs. Ces approches ont montré une efficacité de détecter les fautes surtout celles de courte durée. Cependant, elles peuvent engendrer un taux de faux positifs surtout dans le cas de fautes à longue durée, par exemple le bruit [128].

- Une troisième classe d'approches adoptent les techniques d'apprentissage supervisées pour détecter les fautes des données [123][126]. Ces approches intègrent éventuellement des capacités de traitement probabiliste des données permettant de déduire un modèle normal des données à partir des données d'apprentissage. Néanmoins, elles nécessitent une disponibilité des données correctes et incorrectes pour appréhender le système [127]. Or la disponibilité des données incorrectes n'est pas toujours facile, ces approches ne favorisent pas les applications dynamiques et la détection des fautes en temps réel. En outre, ces

approches nécessitent une intervention humaine pour étiqueter les données pendant la phase d'apprentissage.

Dans ce contexte et dans le but d'avoir un système de détection adapté aux RCSF (léger), précis (taux des faux négatif faible) et/ou bénéfique (possibilité de classification des fautes des données en plus de la détection), nous proposons trois approches de détection des fautes des données. Chacune d'elles a des caractéristiques qui la rendent appropriée pour de nombreux déploiements de RCSF:

- Une approche *à base de règles*. Cette approche se base sur la connaissance des types de mesures pour développer des règles ou des contraintes à satisfaire par les mesures des capteurs. Dans notre approche, nous définissons cinq règles pour détecter quatre types des fautes. Cette approche est économe en énergie et classifie les types de fautes en détectant les fautes de courte et longue durée. En effet, cette approche favorise les différents types de déploiements (dense ou dispersé, plat ou hiérarchique) et les différentes tailles de réseau dont les phénomènes observés présentent, en particulier, une corrélation temporelle.

- La deuxième approche à base de neurones non supervisées. « Self –Organization Maps» (SOM) [132]. SOM a été appliquée pour détecter les fautes dans les réseaux de communication[135][136][137]. Elle a été appliquée également dans les RCSF pour évaluer la position des capteurs [138]. Selon notre étude de l'état de l'art, SOM n'a pas été appliquée dans les RCSF pour détecter les fautes des données issues des capteurs. En effet, SOM est une méthode efficace de visualisation et de classification des données. Elle permet la détection des fautes sans connaissance *a priori* du type de phénomène observé. Elle ne requiert pas un étiquetage préalable des données et donc une connaissance *a priori* de types des fautes. Elle favorise les applications dynamiques qui présentent éventuellement de nouveaux événements et dont les caractéristiques des données normales et anormales peuvent évoluer avec le temps. En outre, SOM n'est pas une solution coûteuse en termes de mémoire et coût de calcul [138]. Ce qui rend SOM une solution générale et très favorable pour les applications de RCSF surtout dans le cas de l'indisponibilité des données pré- étiquetées et le cas où le comportement du système est inconnu avant le déploiement des capteurs.

- La troisième approche est un système hybride fondée sur la combinaison des deux approches précédentes pour réduire les taux des faux négatifs ainsi que pour classifier les types des fautes.

Dans la suite de ce chapitre, nous décrivons notre méthodologie de travail ainsi que la base des données utilisée pour l'étude expérimentale. Nous détaillons les méthodes ainsi que les résultats expérimentaux de chaque méthode.

3.4.Méthodologie de travail

Le processus de détection des fautes s'effectue en trois phases : 1) la phase d'analyse des données, 2) la phase d'apprentissage et 3) la phase de test ou validation.

1. La phase d'*analyse des données* est nécessaire pour comprendre :
   - les types des capteurs et l'environnement du déploiement,
   - la corrélation temporelle entre les données d'un même capteur,
   - la corrélation entre les différents types des capteurs,
   - la répartition géographique ou la topologie du réseau,
   - la corrélation spatiale entre les nœuds voisins,
   - les nœuds ou les capteurs en panne,
   - le modèle des données pour chaque type du capteur,
   - la perte de données, *etc.*

2. La phase d'*apprentissage* est nécessaire pour estimer les paramètres de chaque méthode. Cette phase nécessite le partitionnement de la base en deux parties :
   - les données d'apprentissage ou les données empiriques,
   - les données de test ou les données de validation.

   Les données d'apprentissage doivent refléter correctement les modèles des données (la variation des mesures en fonction du temps, l'intervalle de stabilité, *etc*.). D'où l'importance de sélectionner des capteurs et des jours de mesures fiables pour entraîner le système de détection ou de classification des données. L'apprentissage a été effectué manuellement dans l'approche à base de règles et automatiquement dans l'approche à base de neurones.

3. La phase de *test* ou de *validation* consiste à classifier les données de test et à estimer la performance du système de classification.

3.5.Description de la base de données

Pour expérimenter les méthodes proposées, nous avons utilisé une base réelle de données issues de capteurs hétérogènes. Cette base renferme des données provenant de 54 nœuds « Mica2Dot » du Laboratoire « Intel Berkeley» [123], collectées entre le 28 février 2004 et le 5 avril 2004. Les nœuds sont déployés dans le laboratoire selon la topologie décrite dans la Figure 3-1. Les nœuds fonctionnent en utilisant des batteries assurant leur alimentation autonome. Pendant cette période, une base de 2.3 millions de mesures est collectée par TinyDB[2][125] de la plateforme TinyOS [21]. Les nœuds consistent en des capteurs de température, d'humidité, de luminosité et de voltage. Toutes les 31s il y a acquisition de ces données par chaque nœud du réseau. Les valeurs rapportées par chaque noeud sont formées de huit variables : la date, le temps, la période

[2]TinyDB est un système d'exécution de requêtes sur un réseau de capteurs, intégré à TinyOS. TinyDB a été codé dans la même optique d'économie d'énergie que TinyOS, et ne se contente pas seulement de stocker les données, mais les traite, les filtre, les regroupe, et les envoie sur le réseau périodiquement

ou « epoch », l'ID du nœud (de 1 à 54), la température, l'humidité, la lumière et le voltage. La période est un numéro de séquence croissante monotone. Les données qui ont le même numéro de séquence sont produites par différents nœuds dans le même temps. Les données de la base sont incomplètes. Elles peuvent être perdues ou tronquées (défaillance des nœuds/capteurs, pertes de communication,..). Par exemple, la Figure 3-2 montre le pourcentage de la perte des données de température de chaque nœud. En moyenne la perte des données est de 53 %. La température est exprimée en degrés « Celsius ». L'humidité exprimée en pourcents (0%- 100%). La luminosité est exprimée en « Lux ». Une valeur de 1 « Lux » correspond au clair de lune, 400 « Lux » correspondent à un bureau lumineux, et 100,000 « Lux » correspondent à la lumière du soleil. Le voltage est exprimé en «Volt » (2-3).

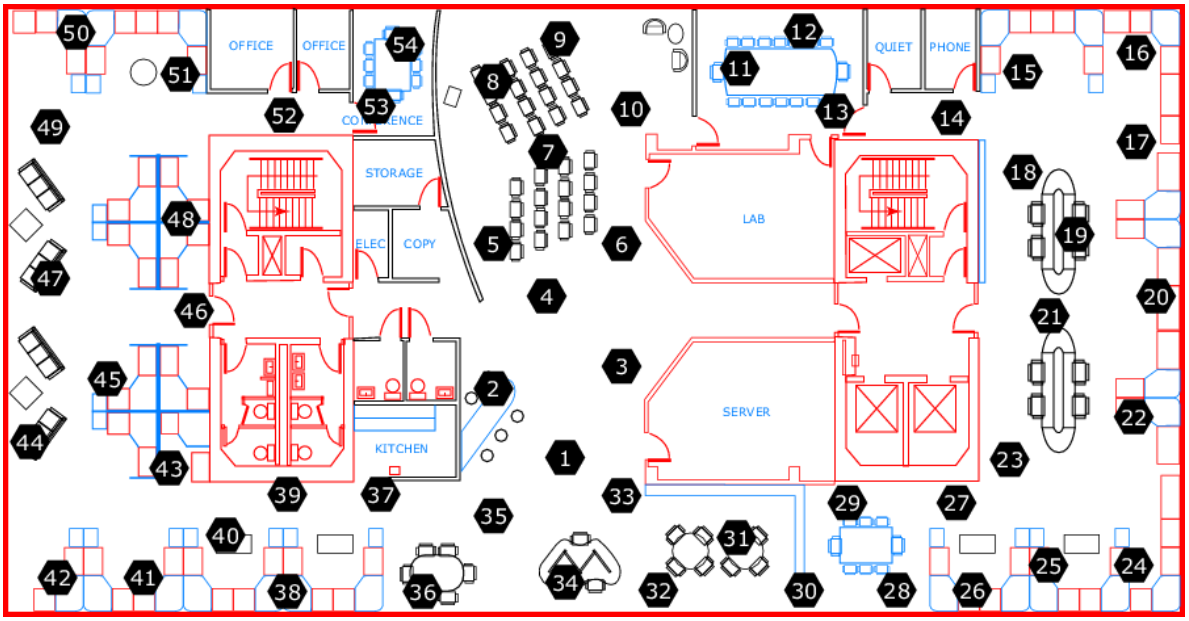


Figure 3-1: Topologie du réseau [123]

3.6.Méthodes à base des règles

Ces méthodes définissent un ensemble de règles pour détecter les fautes des données de capteurs. Les règles de détection des fautes sont appliquées sur des échantillons collectés de chaque nœud du réseau. Un échantillon ou une fenêtre représente un ensemble de valeurs successives du capteur. Nous avons choisi une fenêtre de taille 5 qui favorise, en même temps, le coût mémoire et la précision de la méthode.

3.6.1. Les règles de détection des fautes

Les données issues des capteurs présentent souvent un modèle pour les changements des valeurs entre deux points successifs, un seuil maximal du niveau de bruit tolérable, une stabilité minimale pour les valeurs successives et des bornes supérieures et inférieures pour les valeurs valides [126][128][129]. En utilisant ce modèle des données issues de capteurs, nous avons appliqué cinq règles basées principalement sur la corrélation temporelle pour détecter quatre types de fautes : faute brusque, faute type bruit, faute de blocage et faute due à une valeur aberrante.

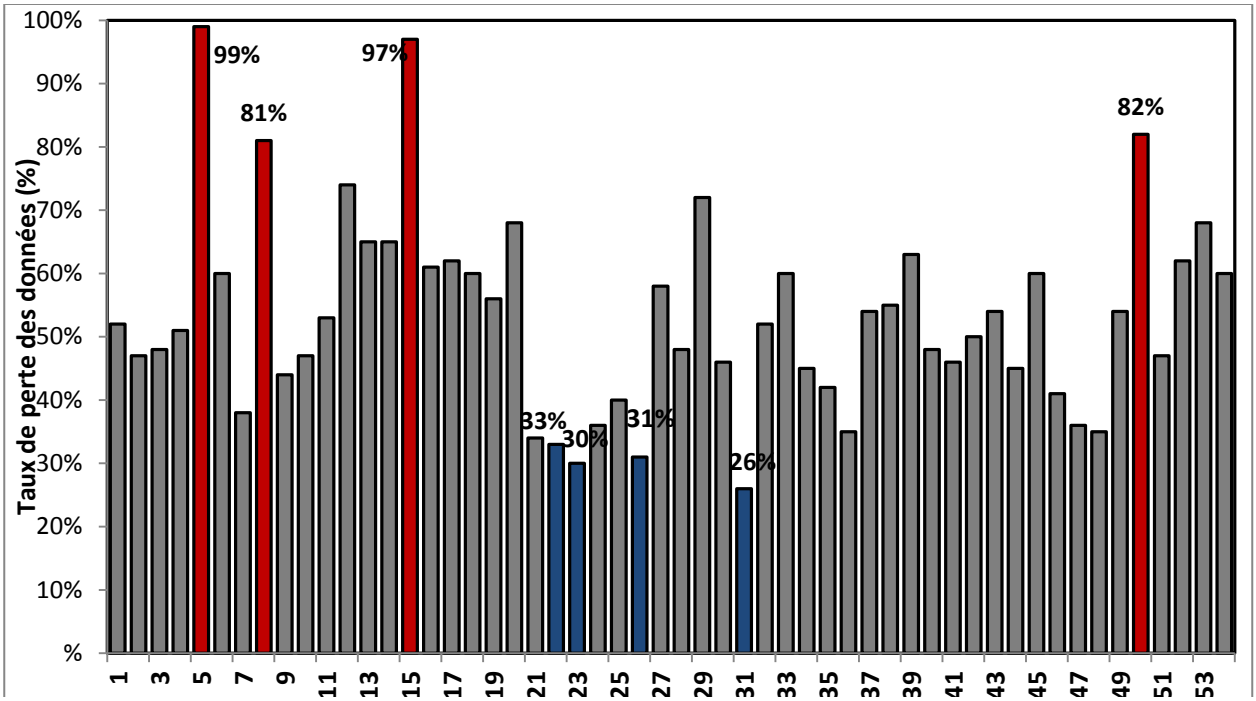


Figure 3-2: Le pourcentage de la perte des données de température de chaque nœud

Règle 1 : Faute brusque

C'est la faute la plus commune et la plus répandue au niveau des fautes de données de capteurs [126]. Les données issues des capteurs suivent souvent un modèle pour le changement des valeurs entre deux points successifs. Une violation significative et soudaine du modèle, indique qu'il y a une faute de type brusque (Figure 3-3). Cette faute est due essentiellement à des défauts matériels, ou des événements produits dans l'environnement, provoquant une discontinuité dans les valeurs rapportées par un capteur. Pour détecter ce type de faute, nous avons calculé le taux de variation entre les moyennes de deux fenêtres consécutives. Considérons $\Delta v, \Delta t \; et \; \Delta max$ qui désignent

respectivement la variation des valeurs, la variation du temps ainsi que le seuil maximal pour la variation des valeurs. La règle de détection est définie comme suit :

$$\frac{\Delta v}{\Delta t} > \Delta max \qquad (3.1)$$

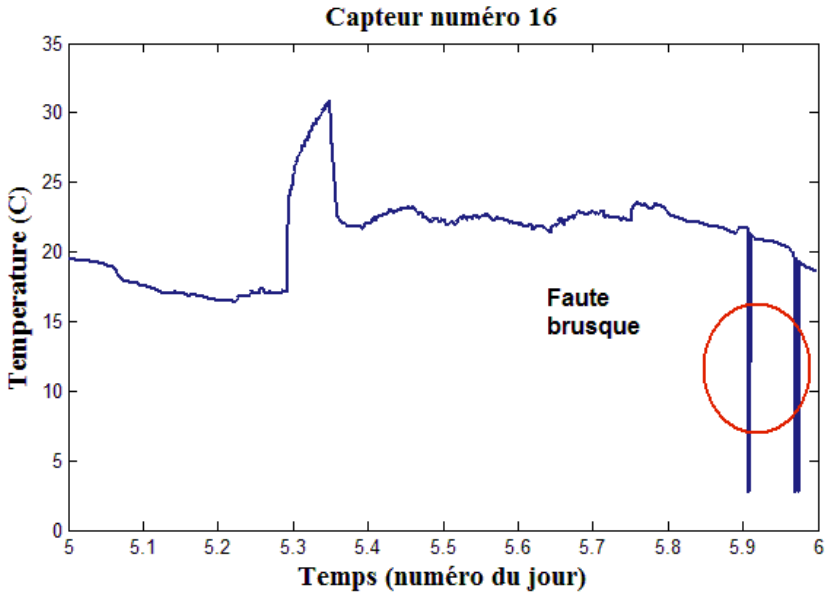


Figure 3-3: Exemple de faute brusque

Règle 2 : Faute type bruit

Bien que le bruit soit répandu dans les réseaux sans fils et prévu dans les données issues des capteurs, un niveau élevé de bruit indique souvent un problème dans le capteur [128]. Un bruit excessif est dû essentiellement aux défauts matériels, et à un manque d'énergie. Une faute de type bruit est caractérisée par une période durant laquelle les valeurs présentent une variation plus élevée que celle prévue (Figure 3-4) Considérons σv, σ max et N désignent respectivement l'écart type de N valeurs successives, le seuil maximum prévu de l'écart –type et la taille de la fenêtre. La règle de la faute bruit est définie comme suit :

$$\sigma v > \sigma max \quad (3.2)$$

Règle 3 : Faute de blocage

Une stabilité minimale est prévue dans un ensemble de valeurs successives rapportées par un capteur. C'est pourquoi, une stabilité anormale indique souvent un défaut dans le capteur (Figure 3-5). Dans ce cas-là, les valeurs rapportées peuvent être très élevées ou très basses en comparaison avec les valeurs normales du capteur. Considérons que σv,σmin et N désignent respectivement l'écart type de N valeurs successives, le seuil minimal prévu de l'écart-type et la taille de la fenêtre. La règle de la faute de type valeur bloquante est définie comme suit :

$$\sigma v < \sigma min \quad (3.3)$$

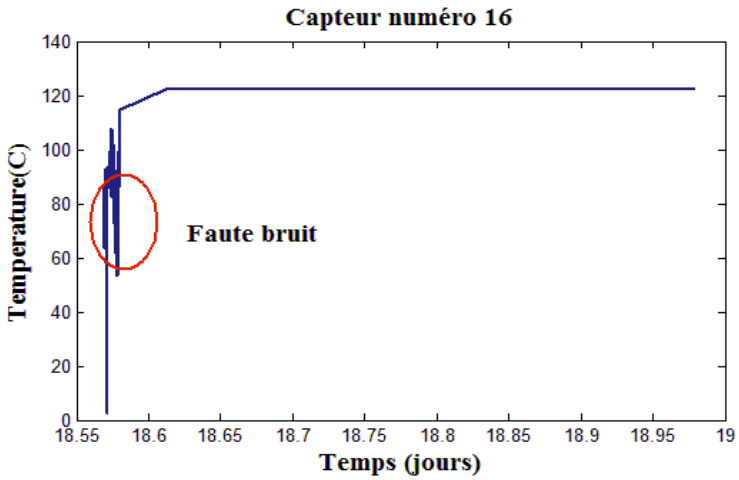


Figure 3-4: Exemple de faute de type bruit

La valeur de σmin peut être obtenue en analysant les mesures, dans un environnement contrôlé, par exemple en laboratoire. La valeur peut être égale ou inférieure à la valeur de la variation minimale observée.

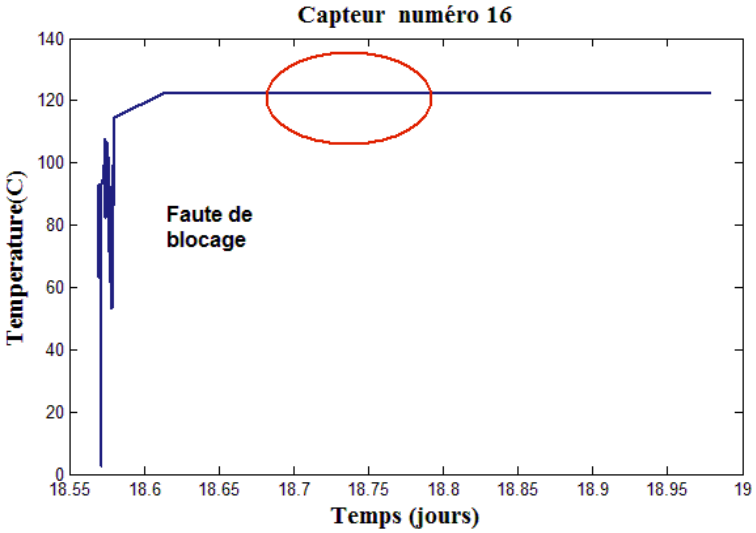


Figure 3-5: Exemple de faute de type blocage

Règles 4-5 : Valeur aberrante

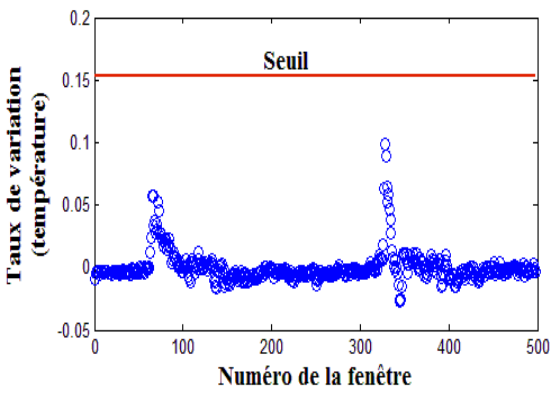
Chaque type de capteur a un intervalle de valeurs valides. Les bornes supérieure et inferieure de l'intervalle dépendent de l'application et des caractéristiques physiques du capteur. Considérons que $\bar{v}, \vartheta max$, ϑmin et N désignent la moyenne des N valeurs consécutives rapportées par le capteur, les bornes supérieure et inférieure de l'intervalle

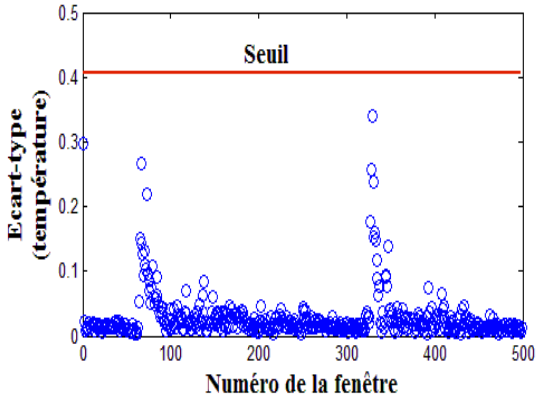
des valeurs valides respectivement. Les règles de la faute de type valeur aberrante sont définies comme suit :

$$\bar{v} > \vartheta max \quad (3.4)$$
$$\bar{v} > \vartheta min \quad (3.5)$$

3.6.2. Détermination des paramètres

Normalement, les valeurs appropriées des paramètres utilisées dans les différentes méthodes de détection de fautes, peuvent être obtenues à travers des connaissances du domaine, la fiche de données du capteur, les modèles de capteurs similaires, l'analyse de l'environnement ou les données historiques sur le comportement du capteur [128][129]. Dans notre approche, nous avons décidé de choisir le nœud le plus fiable comme nœud de référence (Figure 3-6) : dans notre expérience c'est le nœud d'identifiant 31. Ce nœud a un taux de perte le moins élevé parmi les 54 nœuds (Figure 3-2), et présente un modèle de comportement normal (modèle des données semblables) vis-à-vis des autres capteurs (Figure 3-7). C'est pourquoi, nous avons considéré le nœud 31 comme le nœud le plus fiable parmi les nœuds du réseau. En fait, la sélection d'un nœud fiable n'est pas une solution pratique où on peut référer à la documentation des capteurs pour déterminer les valeurs des paramètres. Ici, en absence de la documentation, nous avons déterminé ces valeurs par une phase d'apprentissage hors ligne en se basant sur le nœud 31**.** De plus, nous avons adopté les données des deux premiers jours comme données d'apprentissage, puisque l'état de la batterie au début du déploiement est bon. Dans la Figure 3-6, les bulles représentent les taux de variations entre les moyennes de deux fenêtres consécutives ou les écarts types des fenêtres et la droite horizontale représentent le seuil choisi. Les valeurs des paramètres sont montrées dans le Tableau 3-2. Les valeurs maximales et minimales sont sélectionnées en fonction de la description de la base de données fournie par le laboratoire d'Intel et l'observation faite durant les jours d'apprentissage.





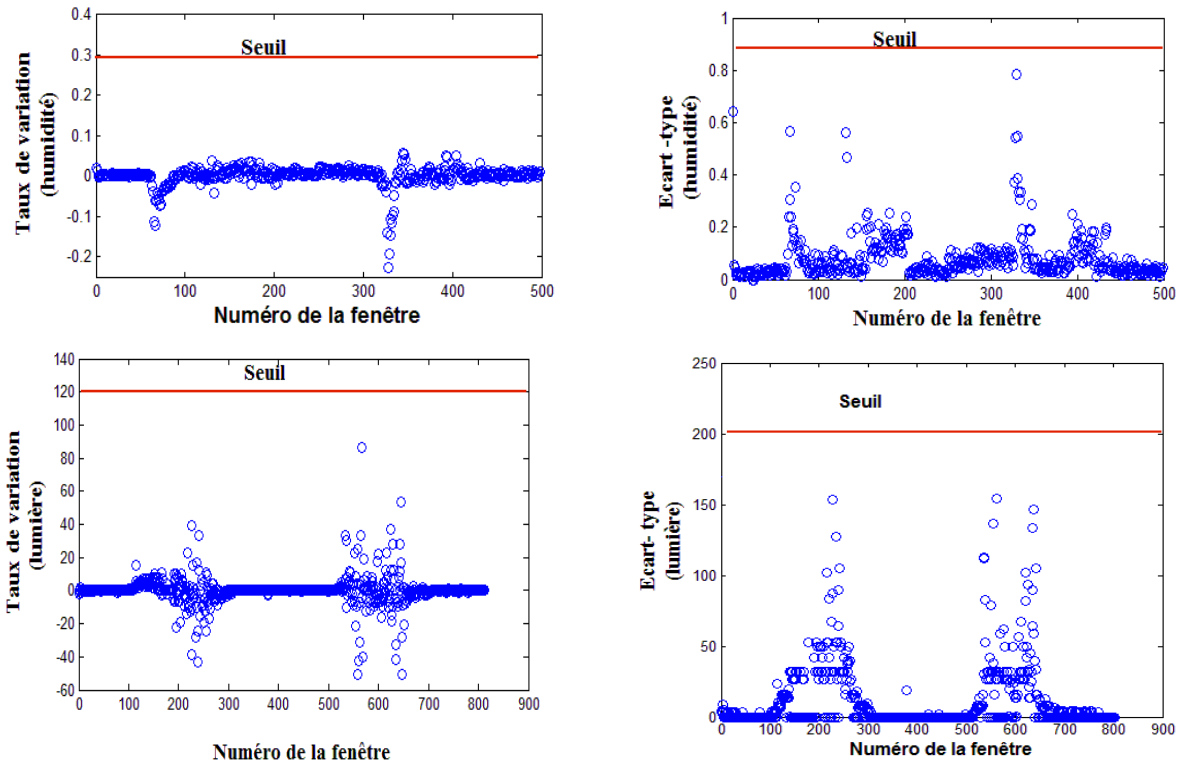


Figure 3-6 : Détermination les valeurs des paramètres des règles de détection

3.6.3. Résultats

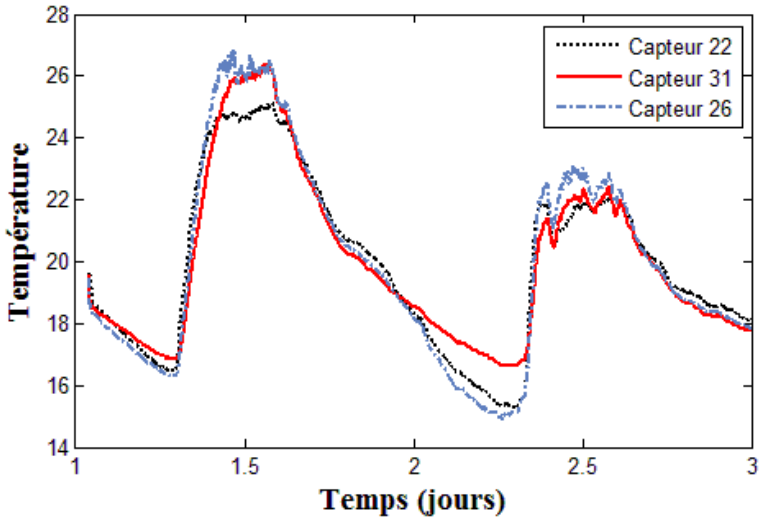


Figure 3-7 : Modèle des données de trois capteurs (22, 31 et 26) pendant les deux premiers jours de déploiement

Pour détecter les quatre types de fautes choisis au niveau des données issues des capteurs, nous avons appliqué cinq règles heuristiques sur la base de données utilisée. Nous observons que les données présentent une combinaison des quatre types de fautes étudiés comme le montre le Tableau 3-1 :

- 20% d'échantillons de température sont détectés comme étant défectueux,
- 17% d'échantillons d'humidité sont détectés comme étant défectueux,
- 21% d'échantillons de luminosité sont détectés comme étant défectueux.

Le pourcentage des fautes de chaque type de capteur est calculé par la formule suivante :

$$Pourcentage\ des\ fautes\ = \frac{Nombre\ d'échantillons\ défectueux}{Nombre\ d'échantillons\ \ total} \tag{3.6}$$

Le taux moyen d'échantillons défectueux a été de 19% du total des échantillons collectés. Ce taux représente la moyenne des pourcentages des fautes de chaque type de capteur. Il est calculé par la formule suivante :

$$Taux\ moyen\ d'échantillons\ défectueux = \frac{\%\ des\ fautes(\ Température)+\ \%\ des\ fautes\ (Humidité)+\%\ des\ fautes(Lumière)}{3} \tag{3.7}$$

Comme l'illustre le Tableau 3-1, nous observons que la faute de type brusque se produit dans un pourcentage le plus faible (1%) en comparaison avec les autres types de fautes. La faute de blocage et de valeur aberrante se produisent dans des pourcentages plus élevés. Le pourcentage de fautes de type blocage est similaire à celui de la valeur aberrante. Cela est dû au fait que les capteurs se bloquent à des valeurs situées en dehors de l'intervalle des valeurs valides. Les capteurs de température se bloquent à la valeur 120 °C, ceux d'humidité a -3.9% et ceux de la lumière à 0.92 Lux.

L'inspection des valeurs de voltage a montré que les échantillons défectueux ont été bien corrélés avec les derniers jours du déploiement durant lesquels la puissance des nœuds était faible (moins de 1.4), comme illustré à la Figure 3-8 . En effet, l'énergie du capteur est un facteur qui peut limiter l'exactitude et la précision des mesures effectuées par le capteur. Comme l'indique l'auteur de [100], si moins d'énergie est disponible au sein du capteur, moins d'exactitude et de précision pourraient être offertes. Ainsi, un capteur avec un temps de vie réduit ne peut pas offrir autant de précision et d'exactitude qu'un capteur débutant son activité.

Pour bien distinguer les nœuds en bon état et ceux défectueux en présence du pourcentage élevé de perte de données (53%), nous avons calculé un ratio que nous avons appelé le taux de débit utile « Useful Data Rate » (UDR). Ce ratio prend en considération le taux de la perte des données «Data Loss Rate » (DLR) et le taux des fautes « Data Fault rate » (DFR) de chaque nœud. Pour cela, le « UDR » de chaque nœud est défini comme suit :

$$UDR_i^t = (100\text{-}\ DFR_i^t)^*\ (100\text{-}\ DLR_i^t) \qquad (3.8)$$

où «UDR_i^t», DFR_i^t»et«DLR_i^t» désignent respectivement le taux de débit utile, taux de la perte des données et le taux de fautes pour le nœud d'identifiant« i » et de type « t ». Le « UDR_i » du nœud i est défini comme suit :

$$UDR_i = \frac{\left(UDR_i^{température} + UDR_i^{humidité} + UDR_i^{lumière}\right)}{3} \qquad (3.9)$$

Dans la Figure 3-9, nous remarquons que les nœuds dont l'identifiant est égal à 31, 22, 21, 48 et 47 sont les nœuds les plus sains (UDR >55%) puisque leur taux de fautes et leur taux de perte de données sont relativement faibles. En revanche, les nœuds dont l'identifiant est égal à 5, 15, 8, 12, et 29 sont les plus défectueux (UDR <25%) car leur taux de fautes et leur taux de perte des données sont relativement élevés.

3.6.4. Evaluation des performances

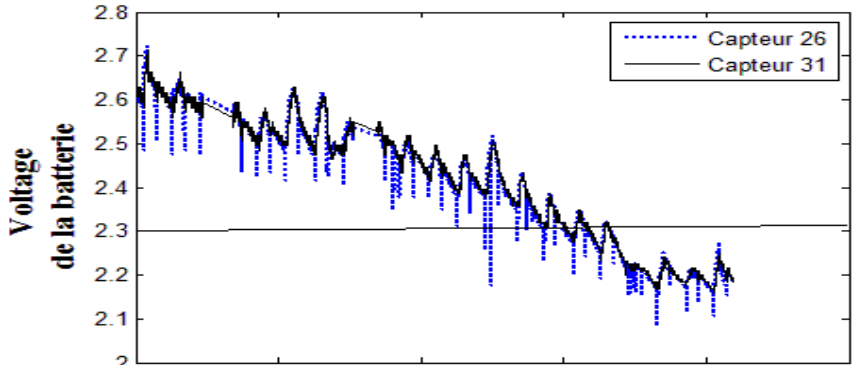
L'approche présentée a été évaluée en utilisant le simulateur TinyOS [21]. Afin d'avoir une solution générique, nous avons implémenté notre approche comme un service configurable sous la couche application. De cette façon, les paramètres de service (Δmax, σmin, σmax, θmin, θmax) peuvent être configurés pour être disponibles à une multitude d'applications et de scénarios. Le service réside sur chaque nœud du réseau. La taille mémoire dépend de la plateforme matérielle. Elle est égale à 1.3 Koctets en ROM et 1.1 koctets en RAM pour la plateforme Micaz (ROM est égale à 128 Koctets et RAM est égale à 4 Koctets). Un autre critère important des performances du service est la consommation d'énergie. Un des principaux avantages de notre approche est son efficacité en termes de consommation d'énergie en effet les nœuds n'ont pas besoin d'échanger des messages pour détecter les erreurs sur les données. Cependant, la base de données adoptée peut contenir des fautes. Dans ce cas, les données d'apprentissage ne peuvent pas refléter correctement les paramètres du modèle de données. C'est pourquoi, la variation des temps a été prise en compte lors de l'application les règles heuristiques. Ce qui permet d'améliorer la précision des résultats en présence d'un pourcentage élevé de perte de données. D'autre part, les méthodes à base des règles sont statiques. Elles peuvent classifier les fautes qui sont définies *a priori* par les règles mais elles ne peuvent pas souvent identifier des fautes de type arbitraire ou des fautes de calibration, *etc*. En fait, ces fautes

peuvent se manifester dans des manières différentes qui rendent leur détection et leur modélisation par les méthodes à base de règles difficiles sans aucune intervention humaine.

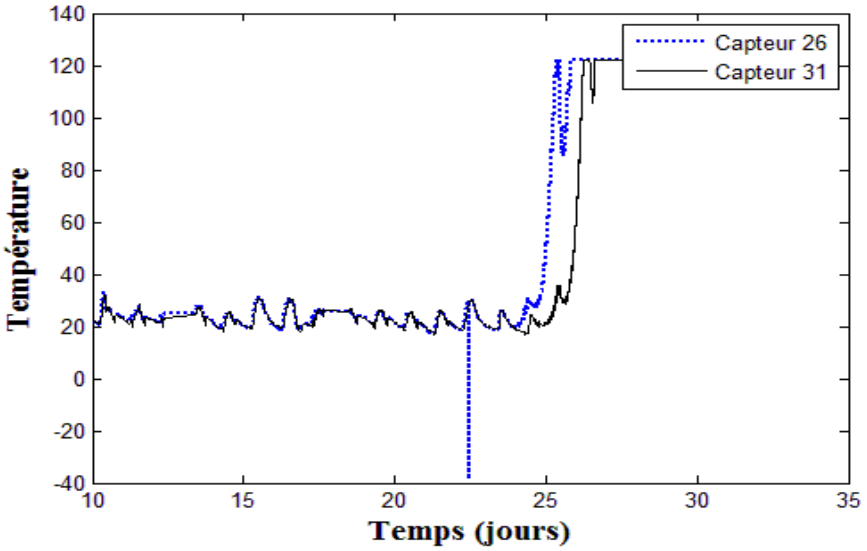
Faute / **Capteur**	Brusque	Bruit	Blocage	Aberrante	**%** des fautes
Température	1%	2%	17%	17%	20%
Humidité	0.5%	1%	15%	13%	17%
Luminosité	1%	2%	18%	10%	21%
Moyenne	1%	1.5%	16.5%	13.5%	19.5%

Tableau 3-1 : Pourcentages des fautes selon les méthodes à base des règles

Paramètre / **Capteur**	Taux de variation $\frac{\Delta v}{\Delta t}$	Ecart-type maximale prevue **σmax**	Ecart–type minimale prevue **σmin**	Valeur maximale prévue ϑmax	Valeur minimale prévue ϑmin
Température	0.15	0.4	0.003	50	-10
Humidité	0.3	0.9	0	100	0
Luminosité	40	100	0	2000	1



(a) Les voltages de deux capteurs 26 et 31. Ligne horizontale représente le niveau de voltage à partir duquel les capteurs commencent à fournir des mesures erronées.



(b) Les valeurs rapportées par les deux capteurs 26 et 31 aux derniers jours du déploiement

Figure 3-8 : Les deux capteurs 26 et 31 rapportent des valeurs erronées (voltage<2.3)

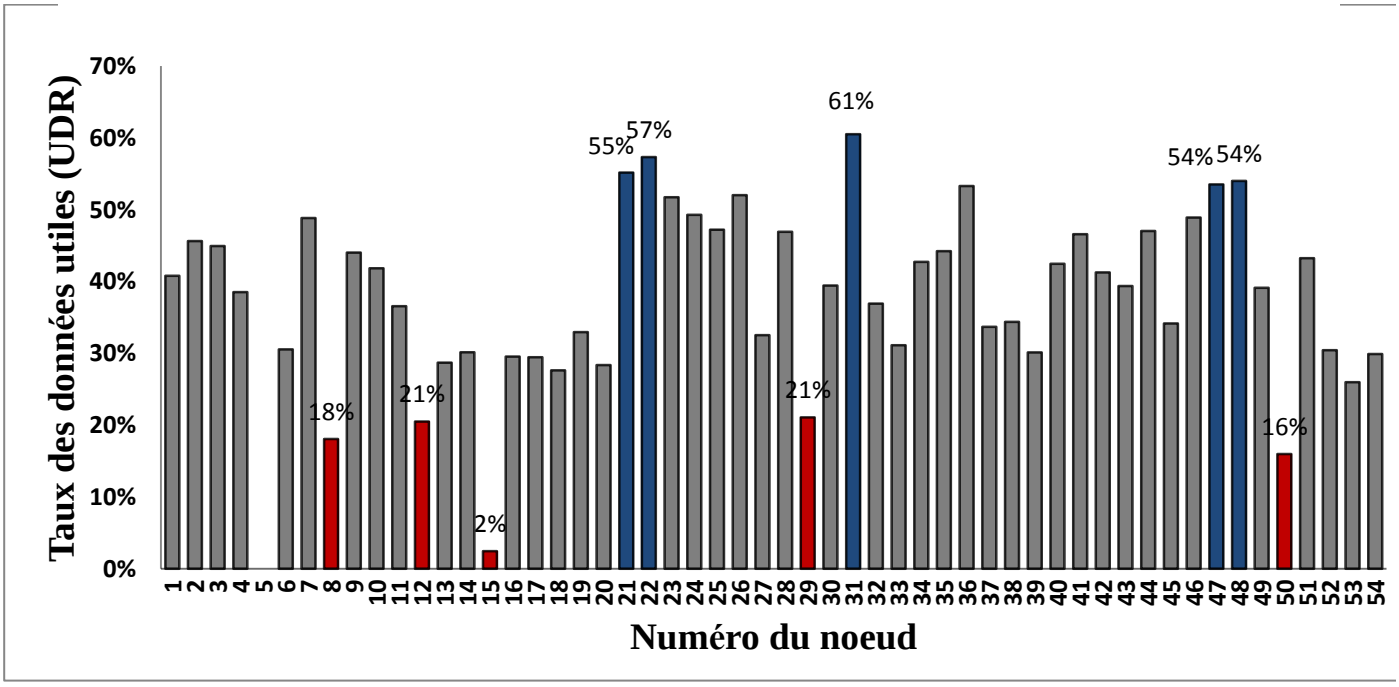


Figure 3-9 : Taux de débit utile des données pour chaque nœud

3.7. SOM- Méthode à base de réseaux des neurones

Les réseaux de neurones sont utilisés pour résoudre de nombreux types de problèmes pour lesquels on dispose de données expérimentales. Ce sont par exemple des

problèmes de reconnaissance de tendances, de classification, de prédiction. Dans notre cas, le réseau de neurones est utilisé pour réaliser la *classification* des données issues des capteurs sans fils en deux catégories : données correctes ou erronées.

En effet, le neurone est conçu comme un automate, doté d'une fonction de transfert, qui transforme ses entrées en sortie selon des règles précises. Par exemple, un neurone additionne ses entrées, compare la somme résultante à une valeur seuil, et répond en émettant un signal si cette somme est supérieure ou égale à ce seuil. Enfin, l'efficacité de la transmission des signaux d'un neurone à l'autre peut varier : on parle de « poids synaptique », et ces poids peuvent être modulés par des règles d'apprentissage.

Les réseaux de neurones se divisent en deux principales classes, les réseaux à apprentissage *supervisé,* et les réseaux à apprentissage *non supervisé*. Pour les réseaux à apprentissage supervisé (par exemple, Perceptron [131]), on présente au réseau des entrées, et en même temps les sorties que l'on désirerait obtenir. Par exemple, on lui présente en entrée une mesure de température " 120 ° C " et en sortie la classe correspondante à l'entrée " donnée erronée ", ce que l'on appelle l'étiquetage des données. Le réseau doit alors se reconfigurer, c'est-à-dire calculer les poids afin que la sortie obtenue corresponde bien à la sortie désirée. Pour les réseaux à apprentissage non supervisé, on présente une entrée au réseau, et on le laisse évoluer librement, jusqu'à ce qu'il se stabilise.

Dans notre étude, nous avons choisi une approche de réseaux de neurones non supervisé : SOM[132]. Notre approche se base sur la corrélation spatiale entre les mesures des capteurs voisins afin de détecter les fautes. L'idée de la corrélation spatiale est illustrée par la droite de la première bissectrice (Figure 3-10) si on représente les données provenant d'un capteur en fonction des données provenant de son voisin et qui doivent être identiques, la droite à obtenir est théoriquement la première bissectrice (Y=X). Cette droite reflète la ressemblance existante entre les données de deux capteurs corrélés spatialement.

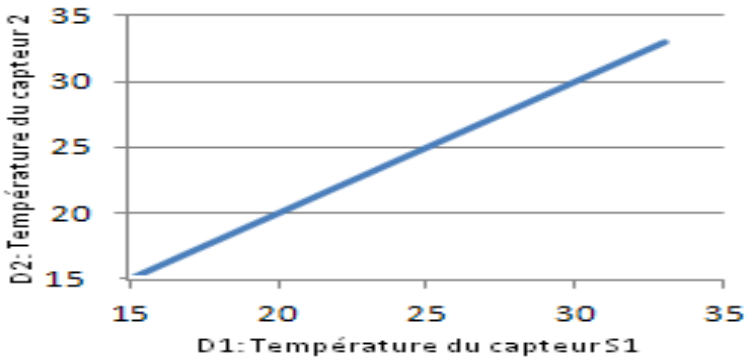


Figure 3-10: Relation théorique entre les lectures des capteurs corrélés spatialement

réseau de neurones, fondée sur des méthodes d'apprentissage non-supervisées. On les désigne souvent par le terme anglais « self- organizing- maps » « SOM ».

Elles sont utilisées pour cartographier un espace réel, c'est-à-dire pour étudier la répartition de données dans un espace à grande dimension. En pratique, cette cartographie peut servir à réaliser des tâches de classification, ce qui nous intéresse dans notre étude. L'avantage principal de l'approche « SOM » par rapport aux approches supervisées est la classification des données selon les points communs qui les rassemblent, sans avoir besoin d'étiqueter les données pendant la phase d'apprentissage. L'application de « SOM » s'effectue en deux étapes : une phase d'apprentissage et une phase de test.

Pour appliquer la corrélation spatiale, nous avons divisé les nœuds du réseau en treize groupes (Tableau 3-3). Les critères de regroupement sont :

- la position géographique des capteurs (surface maximale de 4 m^2)
- Le degré de la correlation entre les mesures des capteurs (coefficient de corrélation supérieure à 0.8).

Nous avons exclus les groupes qui comprennent des nœuds totalement défaillants, (par exemple, les nœuds dont le numéro est égal à 5 et 15) et les groupes dont les nœuds montrent une mauvaise synchronisation temporelle sur les valeurs rapportées par les capteurs.

Groupe	1	2	3	4	5	6	7	8	9	10	11	12	13
	1	4	8	11	14	16	23	29	32	35	38	41	44
	2	6	9	12	18	17	27	30	33	36	39	42	45
	3	7	10	13	19	19	28	31	34	37	40	43	47

Tableau 3-3 : Groupes des nœuds voisins

3.7.1. Apprentissage

L'apprentissage consiste à élaborer la carte des neurones de façon à représenter l'ensemble des données. Chaque neurone se spécialise afin de représenter soit les données correctes soit les données erronées. Nous avons appliqué l'algorithme d'apprentissage avec les paramètres suivants (Tableau 3-4) :

- La *topologie* utilisée est la grille « hexagonale » (Figure 3-11). La topologie définit un voisinage spatial pour chaque neurone de sortie. La forme de ce voisinage pouvant être rectangulaire, hexagonale, aléatoire, *etc.* La Figure 3-11 montre un exemple d'une carte de 32 neurones (neurones représentés par les cercles) reliés par des segments de voisinage hexagonal dont l'avantage est de rendre égales les distances apparentes dans toutes les directions. La *distance* entre les liens est fixée à « LinkDist ». Cette distance reflète le nombre de liens ou les étapes d'un neurone à un autre. En fait, nous avons fixé ces paramètres aux valeurs par défaut définies en MATLAB puisque la

performance du réseau n'est pas sensible au choix exact de la structure topologique [133].

- Le nombre de neurones détermine la granularité des résultats (le nombre de données appartenant à un neurone) et l'expressivité visuelle des classes [134]. Si le nombre de neurones est trop petit (par rapport au nombre des clusters/classes), la carte de neurones sera trop globale et elle pourra cacher des différences éventuellement importantes pour regrouper les données. En revanche, si le nombre de neurones est trop grand, la carte sera trop détaillée et elle pourra identifier des classes différentes pour des données semblables. Ici, on vise à classifier les données en deux classes principales : classe des données erronées et classe des données correctes, puis en quatre sous-classes pour déterminer lequel des capteurs : S1, S2, S3 est défaillant ou produit une erreur inconnue respectivement. Par conséquent, avec un nombre de six classes visé par la carte SOM, nous avons essayé les nombres 16, 32 et 64 neurones. Avec le nombre 64, la carte a été trop détaillée pour représenter les 6 classes visées. Tandis qu'avec une carte de taille 16 et 32 neurones, nous avons obtenus des résultats stables. Dans la suite, nous allons montrer les résultats expérimentaux avec une carte de 32 neurones.

- Le *nombre d'itérations* est fixé à 1000 « epochs », suffisant pour assurer la convergence des neurones.

- La *dimension d'entrée* est fixée à 3. Un système à trois dimensions est souvent possible dans les déploiements de RCSF qui se caractérisent par leur densité des nœuds. Les entrées du réseau de neurones sont les distances entre les données des capteurs.

Topologie	Hexagonale
Nombre de neurones	32
Distance entre les liens	LinkDist
Nombre d'itérations	1000
Dimension des entrées	3

Tableau 3-4 : Paramètres de la simulation

L'étape d'apprentissage est appliquée sur les données des 5 premiers jours. A l'état initial, les poids sont initialisés d'une manière aléatoire. Après l'apprentissage, l'algorithme converge vers une carte organisée qui conserve la topologie de l'espace d'entrée. La Figure 3-12 montre la topologie de la carte avant et après l'apprentissage en dimension 3. Les axes représentent les poids des neurones, les neurones sont représentés

par des bulles qui sont reliées par des segments de voisinage et l'espace d'entrée est schématisé autour de la carte après sa convergence.

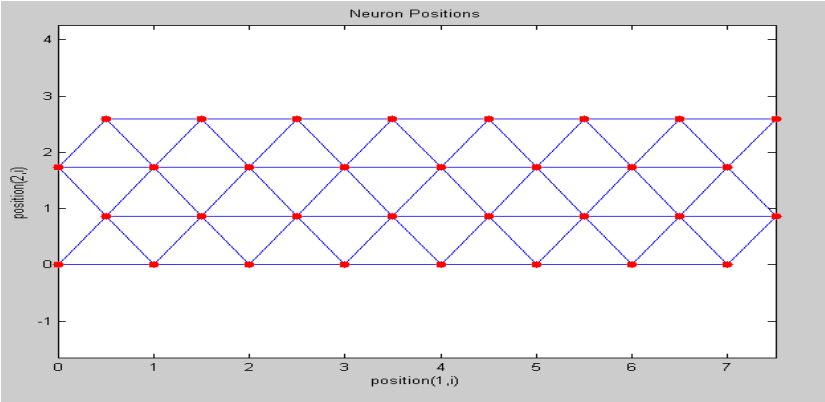
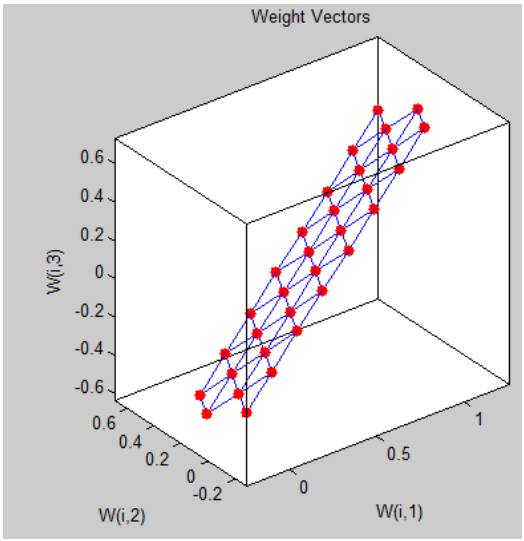


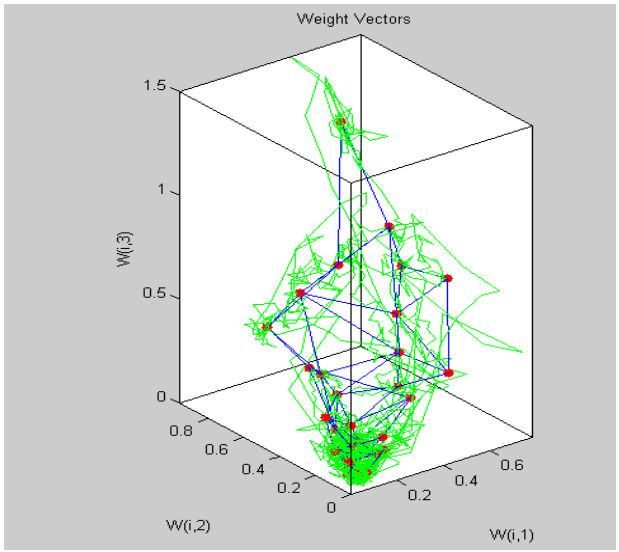
Figure 3-11-Topologie hexagonale de 32 neurones en dimension 2

La classification est faite à base de la méthode des histogrammes en se basant sur la distance séparant chaque neurone de l'origine en deux classes :

- la classe des neurones qui représente les données erronées ;
- la classe qui représente les données correctes.



(a)



(b)

Figure 3-12 - Topologie hexagonale de 32 neurones en dimension 3 : (a) la carte avant l'apprentissage - (b) la carte après la convergence avec la distribution des données.

3.7.2. Test

Cette phase est appliquée pour tous les jours restants. Elle consiste à calculer la distance séparant l'entrée du test (distances entre les capteurs) de tous les neurones du système. Le neurone gagnant est celui dont la distance est minimale. On se réfère à la classe de ce dernier pour juger alors la donnée comme correcte

ou erronée. Les taux d'erreurs de tous les ensembles sont présentés pour tous les capteurs. La moyenne des pourcentages d'erreurs est d'environ 31.5%. Notre approche SOM peut détecter les fautes sans connaissance *a priori* des types de fautes qui peuvent apparaître pendant le déploiement de RCSF. Cependant, elle ne peut pas classifier les fautes comme le font les méthodes à base des règles. En

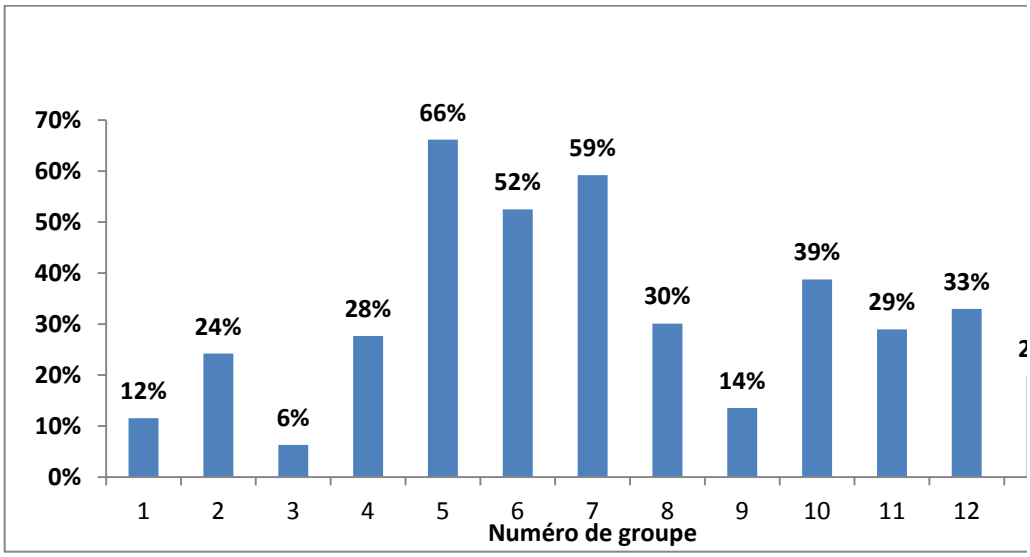


Figure 3-13: Pourcentages des erreurs obtenues par SOM3D

plus, cette méthode ne peut pas détecter les fautes simultanées qui peuvent être dues aux facteurs influant les capteurs situés dans une même zone géographique, par exemple, les effets environnementaux.

3.8.Méthode hybride

La méthode hybride consiste à appliquer les méthodes à base de règles et la méthode à base de SOM afin de réduire le taux des faux négatifs. La méthode considère un échantillon comme défectueux si au moins une de deux méthodes l'identifie comme défectueux. Nous avons considéré les mêmes valeurs des paramètres utilisées lors de l'application de chaque méthode séparément. En appliquant la méthode hybride le taux d'erreur moyen s'élève à 43.5%. En fait, en plus des fautes détectées par la méthode SOM (faute brusque, faute bruit et arbitraires), la méthode hybride détecte les fautes de type blocage et aberrantes qui ne sont pas détectées par la méthode SOM (Tableau 3-5). Par conséquent, la méthode hybride réduit le taux des faux négatifs et classifie les quatre types de fautes.

Cependant, la méthode hybride peut engendrer un taux de faux positifs qui sera égal à la somme des taux de faux positifs des méthodes à base de règles et de la méthode SOM. Le taux de faux positifs des méthodes à base de règles peut être dû à une conception imprécise d'une règle de détection des fautes, à l'impact d'une faute détectée

dans une fenêtre sur les données des fenêtres suivantes, à des changements dans l'environnement, etc. Le taux de faux positifs de la méthode SOM peut être dû par exemple, à un apprentissage imprécis de la carte.

Faute / Approche	Brusque	Bruit	Blocage	Aberrante	Arbitraire
Méthodes à base des règles	+	+	+	+	–
SOM	+	+	–	–	+
Hybride	+	+	+	+	+

Tableau 3-5: Types des fautes détectées par les méthodes à base des règles, SOM et hybride

3.9.Bilan des résultats et discussion

Les travaux expérimentaux montrent que le pourcentage des fautes détectées est égal à :

- 19% pour les méthodes à base des règles,
- 31.5 % pour la méthode SOM,
- 43.5 % pour la méthode hybride.

Etant donné l'utilisation d'une seule et ancienne base de données (données collectées depuis 2004) d'Intel, nos travaux expérimentaux ne suffisent pas pour évaluer précisément l'efficacité des méthodes présentées ci-dessus. En effet, l'indisponibilité d'autres bases de données (manque de matériels, manque de bases téléchargeables à partir d'Internet ...) reflétant divers cas de déploiements réels rend difficile l'élargissement de notre cadre d'expérimentation et par suite l'évaluation de nos résultats. Cependant, en se basant sur l'analyse des résultats obtenus, l'observation des données et notre maîtrise des techniques appliquées, nous pouvons tirer les conclusions suivantes :

Les méthodes à base de règles peuvent détecter quatre types de fautes en appliquant cinq règles basées principalement sur la corrélation temporelle. Cette approche présente de nombreux avantages, parmi lesquels : le passage à l'échelle, l'indépendance du type de déploiement et la classification des fautes.

En revanche, l'approche à base de règles présente des inconvénients, parmi lesquels : l'efficacité de l'approche est fortement liée au choix des paramètres, la possibilité de non-détection de fautes arbitraires (indéfinies a priori) et la possibilité d'engendrer un taux de faux négatifs.

Pour illustrer ce dernier point, prenons l'exemple d'un départ de petit feu dans le laboratoire d'Intel. Suite à cet événement, la température peut s'élever à une valeur hors du seuil spécifié par les règles du système de détection, par exemple 60 ˚ Celsius. Dans ce

cas-là, le système à base de règles considère les mesures de 60˚ Celsius, rapportées par un ensemble de capteurs, comme fausses même si en réalité ce sont des mesures correctes (effet des événements rares ou environnementaux).

En effet, l'impact des conditions environnementales est plus répandu dans les déploiements extérieurs. Par exemple, un capteur placé dans le périmètre d'une montagne peut collecter des mesures de température vraiment basses, hors du seuil spécifié. Au premier regard, la tendance à dire que le capteur est mal calibré est inévitable. Mais cet effet peut être évité, si on considère que le capteur est déployé dans une zone dans laquelle une tempête de neige a frappé, et que ce dernier se trouve deux mètres en-dessous de la neige. Effectivement, le capteur fonctionne correctement, mais les conditions environnementales ne sont pas celles prévues et affectent les mesures. De multiples problèmes peuvent aussi être issus d'attaques malveillantes ou engendrés par des animaux prés de la zone de détection. En effet, un système à base de corrélation spatiale peut résoudre l'impact des conditions environnementales en comparant les mesures des capteurs situés dans la même zone géographique.

La méthode SOM est une méthode de classification dynamique des données. Le principal avantage de notre approche est la détection des fautes sans connaissance *a priori* du phénomène observé et des types de fautes. En revanche, l'inconvénient principal de notre approche à base de corrélation est la non-détection de fautes simultanées.

Pour illustrer ce point, prenons l'exemple des derniers jours de déploiement durant lesquels les capteurs fournissent la même mesure erronée (température égale à 120˚ Celsius), sous l'impact de l'affaiblissement de la puissance de la batterie. Le système de détection à base de corrélation spatiale ne considère pas une température de 120˚ Celsius comme une erreur car cette valeur est rapportée simultanément par plusieurs capteurs voisins

La méthode hybride fondée sur deux méthodes présente de nombreux avantages, parmi lesquels : la détection des fautes simultanées, la possibilité de détection des fautes arbitraires (inconnues *a priori*) et la classification des fautes.

En revanche, l'inconvénient principal de cette méthode est la possibilité d'engendrer des taux de faux positifs.

3.10. Conclusion

Dans l'optique de concevoir une approche globale de la gestion des fautes adaptée aux RCSF, nous avons traité la problématique de la qualité des données issues de capteurs d'une manière théorique et expérimentale.

Dans la première approche théorique, nous avons étudié les travaux de recherche qui analysent les aspects de l'exactitude, la précision et la fiabilité des données. Nous avons vu que les données des capteurs sont souvent hétérogènes, collectées dans des situations

souvent difficiles, possèdent des caractéristiques spatiales et temporelles, et évoluent au cours du temps. Ces spécificités nécessitent donc un traitement particulier en terme de qualité prenant en considération l'hétérogénéité et la dynamique des données capteurs. L'étude théorique a été conclue par une proposition de trois méthodes fondées sur des techniques de traitement du signal qui sont :

- Méthode à base de règles fondée sur la corrélation temporelle,
- Méthode d'apprentissage automatique à base de réseaux de neurones.
- Un système hybride fondé sur la combinaison des deux méthodes précédentes.

Dans la partie expérimentale, nous avons appliqué les méthodes sur une base de données du monde réel. La méthodologie de notre étude consiste à :

- analyser la base de données,
- estimer les paramètres de chaque méthode sur des données d'essai,
- valider l'efficacité de chaque méthode sur le reste des données.

Finalement, chacune de ces méthodes présente des avantages et des limites selon le contexte de déploiement. Une approche hybride fondée sur la méthode à base de règles et l'approche d'apprentissage non supervisée basée sur des critères de classification spatiale améliore le taux de détection des fautes et respecte les enjeux de la qualité des données issues de capteurs exploitables dans des applications de RCSF.

Chapitre 4.

DIAGNOSTIC DES RESEAUX DE CAPTEURS

Le diagnostic vise à obtenir un compte-rendu des défaillances, des risques et des coûts inutiles, se rapportant au fonctionnement du système. Un outil efficace de diagnostic, aide l'administrateur du système, à déterminer le plus précisément et le plus précocement possible, les éléments qui doivent être réparés. Dans ce chapitre, nous présentons le service de diagnostic qui fait partie de notre approche globale de la tolérance aux fautes.

SOMMAIRE

4.1.Introduction

Dans les RCSF, la défaillance se manifeste à plusieurs niveaux (nœud, réseau, données, et application) et peut parfois se propager dans l'ensemble du système en dégradant la qualité des services fournie à l'utilisateur final (Figure 4-1). Par exemple, l'épuisement de la batterie d'un nœud provoque sa défaillance et par conséquence, elle provoque la perte des messages des nœuds descendants de l'arbre de routage. De la même façon, si la couche finale (la couche applicative du système) présente des fautes logicielles et/ou matérielles, l'ensemble du système sera en panne.

Mettre en œuvre un système de diagnostic des défaillances est alors nécessaire pour déterminer l'élément qui doit être réparé. La précision du diagnostic favorisera l'efficacité des opérations de maintenance. Par exemple, la détection et la localisation d'un défaut au moment de son apparition permet de réparer rapidement le composant incriminé avant que son impact ne devienne critique. De ce fait, l'arrêt du système complet engendrant des coûts de maintenance élevés peut-être évité. En revanche, une mauvaise mise en œuvre d'une solution va entraîner non seulement une perte de temps, une diminution de la bande passante et de l'énergie, mais peut aussi aggraver le problème, en dégradant les performances du système.

La mise en œuvre du système de diagnostic dans un RCSF pose donc de nombreux défis en termes de qualité de service attendue par l'administrateur, parmi lesquels : le faible coût surtout en termes d'énergie, la transparence, l'adaptabilité**,** la détectabilité, l'isolabilité (précision), les aspects temps réel, le passage à l'échelle, l'extensibilité,

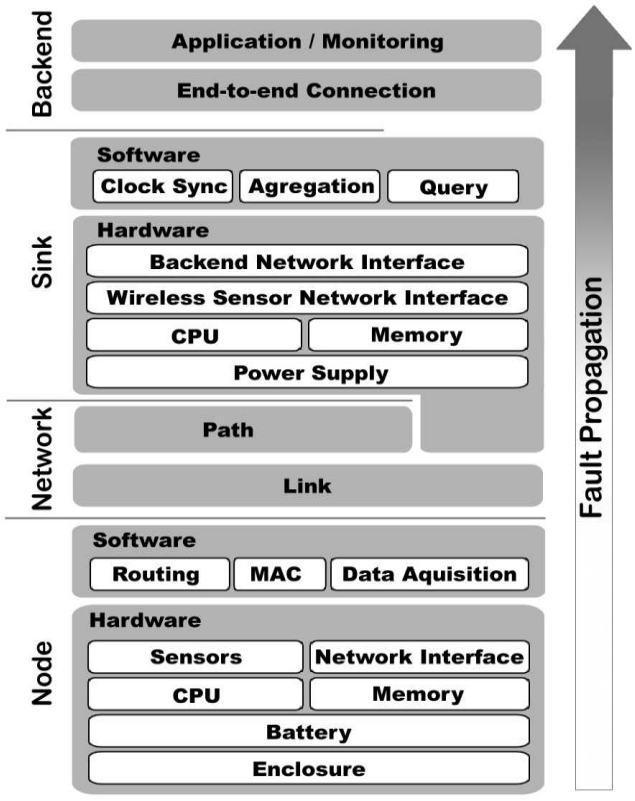


Figure 4-1: Propagation des fautes dans le RCSF [137]

l'interopérabilité, *etc*. Avec une source d'énergie définie, tous ces paramètres de performance ne peuvent pas être optimisés simultanément. En effet, les besoins de Qualité de Service (QdS) sont différents d'une application à l'autre. Par exemple, les aspects temps réel sont prépondérants pour les applications de sécurité critiques. De ce fait, le système de diagnostic doit avoir un support pour la qualité de service en fonction des besoins de l'application. Le but de notre travail est donc de relever le challenge de diagnostic en répondant, le plus efficacement possible, aux exigences des applications en termes de (QdS).

4.2. Positionnement

Les approches de diagnostic présentées dans la littérature sont nombreuses (voir le chapitre de l'état de l'art). Chaque approche se caractérise de la façon dont elle gère les différents critères de QdS.

- Une classe d'approches inspecte l'état interne du système en envoyant des commandes/requêtes pour explorer des variables ou des paramètres au niveau du nœud[60][61]. Ces approches permettent la visibilité de l'état interne du nœud. Cependant, la gestion à distance des nœuds ne permet pas le passage à l'échelle.

- Une deuxième classe d'approches, considère le diagnostic au niveau du trafic des paquets [70][140]. Cette classe, introduit principalement des unités de renifleurs, pour capter et analyser le trafic du réseau. Le diagnostic passif est transparent, il évite l'interférence avec les opérations du système surveillé, au détriment du coût des matériels supplémentaires. En outre, le domaine des défaillances d'une telle classe est limité aux problèmes qui se manifestent uniquement sur le trafic des paquets.

- Une troisième classe consiste à analyser les défaillances, d'une manière centralisée à travers des informations recueillies périodiquement, auprès des nœuds du réseau [55][56]. Le domaine des défaillances des approches centralisées est suffisamment large pour analyser des problèmes globaux comme par exemple, le partitionnement du réseau. Cependant, cette catégorie de solutions est tributaire de la consommation d'énergie et ne permet pas le passage à l'échelle.

Pour dépasser les limitations des approches ci-dessus, nous adoptons une approche distribuée à base d'arbre de décision associée à la fonctionnalité de marquage des paquets (Figure 4-2). En effet, le diagnostic distribué résout le problème de la dissipation d'énergie des approches centralisées. Le marquage des paquets résout le problème de visibilité limitée du diagnostic passif. Similairement à de nombreuses approches distribuées [86][141][142][143], l'approche adoptée utilise un algorithme de détection à deux phases afin d'identifier les défaillances. Cependant, notre approche, nommée « SMART » permet de réaliser les compromis de (QdS) en fonction des besoins de l'application, indépendamment du type de l'application et du scenario du déploiement du système. Nous bénéficions de l'approche orientée-service pour contrôler le fonctionnement du service en fonction les exigences de l'application en termes de

robustesse, de la consommation d'énergie, de la latence de détection, de la perte tolérable de paquets, *etc*.

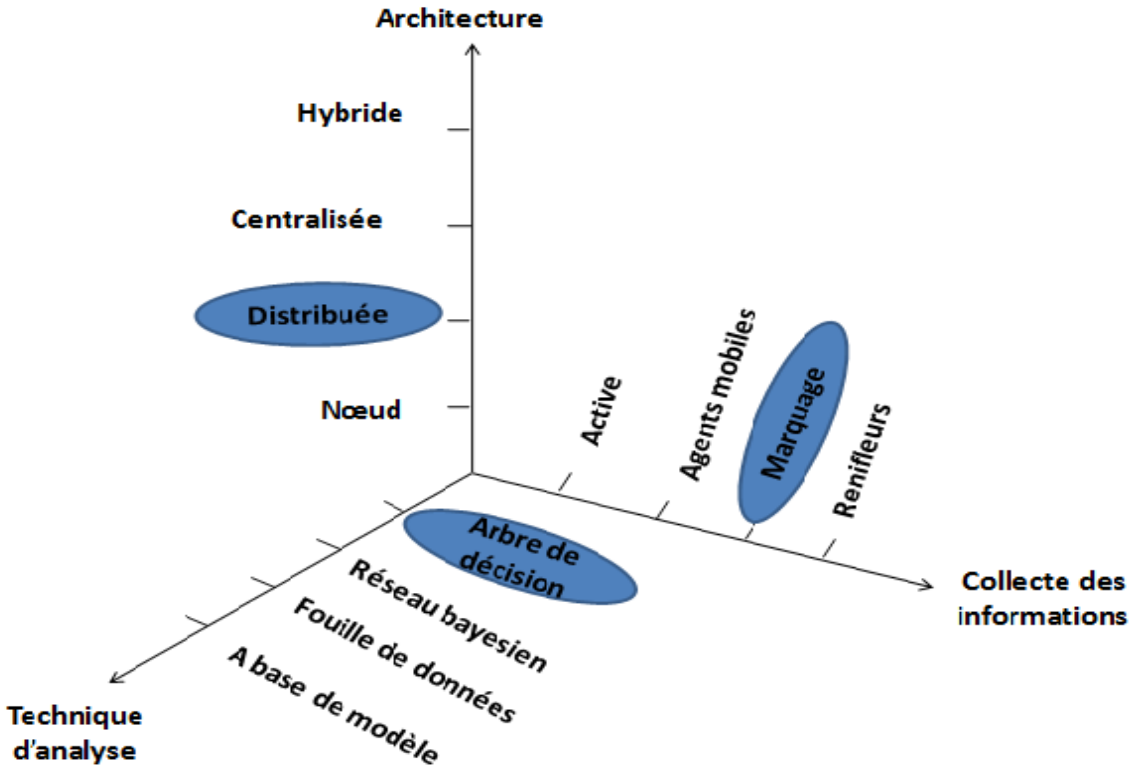


Figure 4-2 : Critères retenus pour le service de diagnostic

4.3. Description du service proposé SMART (Self Monitoring Adaptative and Ressource efficienT service)

Bien que la connectivité et les fonctionnalités des réseaux de capteurs aient été améliorées par de nombreux travaux [144][145][146][147], les réseaux de capteurs souffrent encore de nombreux problèmes liés aux nœuds et aux liens du réseau. Ici, le service diagnostique deux types de défaillances qui sont répandus dans les RCSF [148] :

- La défaillance d'un nœud due à l'épuisement de l'énergie,
- La défaillance des liens due à une mauvaise connectivité.

Lorsqu'une batterie d'un nœud se vide, on dit qu'il est en panne puisqu'il n'est plus dans la capacité d'interagir avec les autres nœuds du réseau. Un nœud prend une décision locale, basée sur des suppositions concernant l'existence de liens de communication distants. La non-existence d'un lien ou la mauvaise connectivité après déploiement peuvent être assimilées à une panne de lien de communication. Le service effectue la

tâche de diagnostic, tout en gérant les cycles d'activité du nœud et le nombre de voisins à surveiller. Le diagnostic s'effectue en trois étapes :

- la détection des éléments défaillants,
- la détermination des causes des défaillances (diagnostic),
- la notification des rapports de défaillances.

En effet, le service proposé SMART met l'accent sur le diagnostic des causes principales de la réduction du débit des données (ce problème est de premier ordre dans les déploiements des RCSF). Les causes profondes de la réduction du débit des données ont été attribuées en grande partie à la défaillance des nœuds et/ou liens qui présentent un comportement asymétrique, irrégulier et/ou variant dans le temps [148].

4.3.1. Caractéristiques

Le service SMART est adapté aux systèmes de RCSF puisqu'il fournit les fonctionnalités de diagnostic d'une manière générique, adaptative et efficace.

Service générique. SMART diagnostique le système indépendamment des éléments suivants :

- type d'application (orienté événements, orienté données ou orienté requêtes),
- la topologie du réseau (plate ou hiérarchique),
- type de déploiement (intérieur, extérieur),
- densité du réseau (dense ou dispersé)
- et des protocoles des couches MAC et routage.

Du point de vue conception, le service est implémenté dans une couche indépendante de la pile protocolaire (Figure 4-3), mais facilement intégrée avec les couches de la pile. Il fournit aux applications de nombreux paramètres ajustables, qui le rendent approprié pour de nombreux types de RCSF. En ajustant les paramètres, l'application peut réaliser le compromis requis entre les différents enjeux de diagnostic : la latence de détection, la consommation énergétique, la robustesse, la perte de paquets tolérable, le taux de fausses alertes, *etc*.

Service adaptatif. Les réseaux de capteurs se caractérisent par une forte dynamique de la topologie surtout lorsqu'il s'agit d'un réseau mobile. La moindre défaillance énergétique d'un capteur, peut changer significativement la topologie du réseau. La perturbation des communications (comme les obstacles, l'interférence, *etc*.) peut induire des cassures de liens entre les nœuds voisins. L'administrateur peut ajouter, déplacer ou enlever des nœuds à n'importe quel moment selon les besoins de l'application. Par exemple, l'emplacement des nœuds peut s'avérer inadéquat pour récupérer les données requises. De nouvelles zones d'intérêts peuvent également apparaître, rendant nécessaire l'installation de nouveaux nœuds. En outre, le redéploiement et l'ajout d'autres nœuds, peuvent être envisagés pour pallier quelques défaillances. Dans tous les cas, le service

peut s'adapter aux changements topologiques en gérant le nombre de voisins surveillé par chaque nœud d'une manière dynamique.
Service efficace les résultats expérimentaux montrent que le service, gère efficacement les ressources, tout en fournissant un taux de détection et une précision de diagnostic satisfaisants.

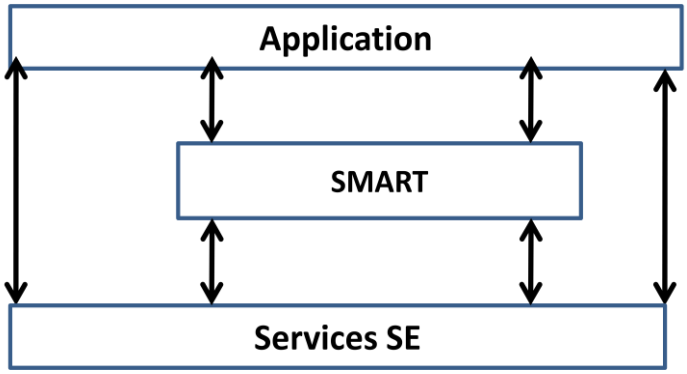


Figure 4-3: Architecture logicielle du service SMART

Nous allons décrire les fonctionnalités du service, le format des messages, l'algorithme de détection et l'algorithme de diagnostic.

4.3.2. Fonctionnalités

Le service fournit plusieurs fonctionnalités permettant de réduire le taux des fausses alertes et d'économiser la consommation des ressources. Ces fonctionnalités sont : la détection des défaillances, le contrôle des cycles d'activité et la gestion des voisins.

4.3.2.1. Détection en deux phases

La détection s'effectue par échange de messages permettant de prendre en compte les pannes ou l'apparition de nouveaux nœuds. Cependant, pour réduire le taux de fausses alertes, le service utilise un algorithme de détection en deux phases: *la phase de détection locale* et *la phase de consensus*

- Pendant la phase de *détection locale*, chaque nœud surveille ses voisins (ou un certain nombre de voisins), et attend pendant une certaine période de temps, un message de type "Heartbeat" provenant de chaque nœud voisin. Pour éviter l'impact des défaillances transitoires, la période doit être suffisamment grande pour permettre la transmission d'un certain nombre de messages de chaque voisin sélectionné. Nous désignons par L ce nombre de messages. Ainsi, seulement lorsque les L messages (et non plus un seul message [141]) sont manqués à partir d'un nœud voisin on le considère comme un nœud suspect (Figure 4-4).

- Pendant la phase de *consensus*, le nœud surveillant échange ses conclusions avec les autres voisins. Cette phase est particulièrement importante pour diminuer les taux de fausses alertes. La période de la deuxième phase doit être aussi suffisamment longue pour permettre l'échange de *C* messages (et non plus un seul message), avant de décider des jugements définitifs et les envoyer à la couche application locale. Les valeurs de L et C ainsi que les valeurs des autres paramètres peuvent être configurés par la couche application.

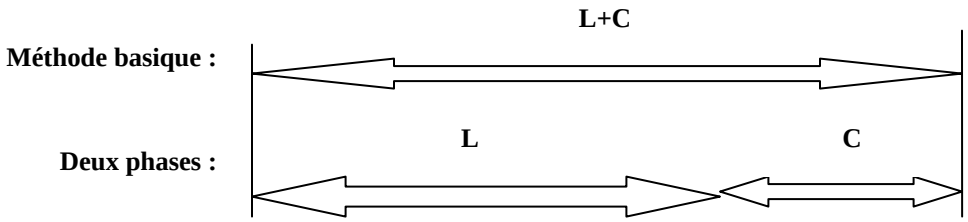


Figure 4-4 : Détection en deux phases

4.3.2.2. Contrôle des cycles d'activité

Pour optimiser la consommation d'énergie, le contrôle des cycles d'activité ou la technique de « duty-cycle » est largement utilisée dans les systèmes à base de RCSF [149][150]. La technique consiste à mettre la radio de l'émetteur en mode veille, à chaque fois que la communication n'est pas nécessaire. Ainsi, les nœuds alternent entre périodes actives et sommeil en fonction de l'activité du réseau (Figure 4-5). Ce comportement est généralement dénommé « duty-cycling ». Un Duty-cycle est défini comme étant la fraction de temps où les nœuds sont actifs. Au début de chaque période active, un nœud démarre sa radio et communique avec ses voisins. À la fin de la période active, le nœud arrête la radio pour le temps restant (période de sommeil) avant la période de cycle de service suivante. Comme les nœuds-capteurs effectuent des tâches en coopération, ils doivent coordonner leurs dates de sommeil et de réveil. Ce qui rend possible l'échange de paquets, même si les nœuds ont un faible duty-cycle (i.e., ils dorment la plupart du temps). Ici, les temporisateurs des cycles d'utilisation radio sont des paramètres du service. Ils peuvent être configurés en fonction des besoins de l'application en termes de latence de détection et de consommation d'énergie. Ainsi, un faible duty-cycle se traduit par une augmentation du temps de latence et une réduction de la consommation d'énergie et vice-versa. Un cycle d'activité élevé favorise les applications critiques.

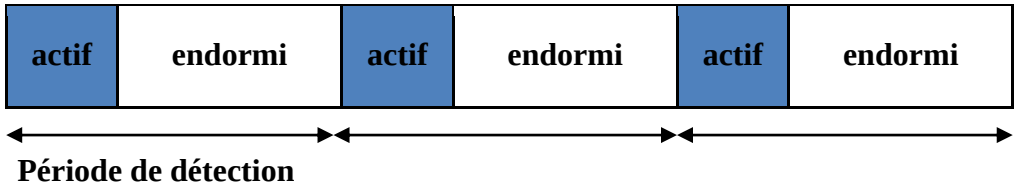


Figure 4-5 : Cycle d'activité

4.3.2.3. Gestion des voisins

Normalement, chaque nœud du réseau maintient une table de voisinage qui répertorie les informations des nœuds situés dans son voisinage. Cette table est construite et maintenue à jour en écoutant de manière passive les communications environnantes. Dès que le nœud souhaite transmettre un message vers un nœud distant, il pourra envoyer un message unicast vers l'un des voisins actuellement répertoriés dans sa table de voisinage. Ce dernier se chargera par la suite de relayer ce message jusqu'à la destination en fonction du protocole de routage utilisé dans le réseau.

La sélection du voisin se fait ici en fonction de la qualité du lien. A la réception d'un paquet, le récepteur mesure la qualité du lien avec le nœud émetteur. Si la qualité d lien est supérieure à un certain seuil spécifique, supposons (*LQmin*), l'émetteur sera considéré comme un nœud voisin. De cette façon, la table de voisinage est construite et maintenue à chaque période d'activité du nœud. La structure et la taille de la table de voisinage sont aussi étudiées. Chaque entrée de la table correspond à un nœud voisin. Une entrée est constituée de l'identifiant, des compteurs *L* et *C*, du niveau d'énergie résiduel, et de la qualité du lien correspondant au nœud voisin. Concernant la taille de la table, deux questions importantes se posent ici :

- Combien de nœuds doivent surveiller un nœud donné ?
- Quel sera le nombre optimal dans le cas d'un réseau dense et dans un réseau dispersé ?

En effet, le nombre optimal n'existe pas [151]. Le nombre minimum (*Nmin*) doit éviter la présence de nœuds isolés, dits nœuds orphelins. Le nombre maximum *(Nmax)* doit établir un compromis entre la consommation des ressources et le degré de robustesse. Plus *Nmax* est élevé, plus la consommation d'énergie est importante et la robustesse est élevée. Ce compromis est lié aux exigences de l'application en termes de (QdS). La robustesse est prioritaire pour les applications critiques. C'est pourquoi, les nombres minimum et maximum de voisins sont considérés ici comme des paramètres du service. Ils peuvent être configurés au début du déploiement, par la couche application, puis ils peuvent être maintenus pour s'adapter à la nature dynamique du RCSF. En effet, la reconfiguration se fait d'une manière dynamique, où chaque nœud ajuste le seuil minimum de qualité du lien (*LQmin*) en fonction de la taille courante de sa table de voisinage. Si le nombre de voisins désiré par l'application est 6, et si le nombre actuel des voisins est 3, donc le nœud diminue le (*LQmin*) pour augmenter le nombre de voisins et vice versa.

4.3.3. Les messages du protocole

Un échange de messages est nécessaire pour surveiller et vérifier l'état d'un nœud avec ses voisins. Le protocole proposé se base sur l'échange de quatre types

de messages. Le format de chaque message dépend de son rôle (Figure 4-6). Mais, tous les messages transmis incluent le niveau d'énergie restant du nœud.

1. Message de « Hearbeat » (H) : message diffusé par chaque nœud pour notifier sa présence aux nœuds voisins. A la réception du message (H), le nœud récepteur met à jour l'entrée de la table de voisinage correspondant au nœud émetteur (réinitialisation du compteur L, mise à jour du niveau d'énergie et de la qualité du lien, *etc.*). La taille du message (H) est de 8 octets. Le message (H) est un message de diffusion à un saut. Il contient les champs suivants :
    - l'identifiant du nœud émetteur,
    - le niveau d'énergie restant,
    - le compteur de messages.

2. Message de requête (Q) : message diffusé par un nœud pour s'informer de l'état d'un nœud suspect. A la réception du message de requête, le récepteur envoie un message de rejet (R) si le nœud sujet de l'interrogation est dans la phase de détection locale. La taille du message requête (Q) est de 10 octets. Le message (Q) est un message de diffusion à un saut. Il contient les champs suivants :

- l'identifiant du nœud émetteur,
- le niveau d'énergie restant,
- le compteur de messages,
- l'identifiant du nœud suspect.

3. Message de rejet (R) : message diffusé par un nœud pour indiquer qu'un nœud suspect est encore en bon état. A la réception du message de rejet, le nœud récepteur compare la valeur du compteur *L* inclus dans le message, à celle mémorisée dans la table des voisins, puis il sauvegarde la valeur la plus grande des deux. Cela permet de réduire la latence de détection, et ainsi d'éviter d'entrer en boucle. La taille du message (R) est de 13 octets. Ce message est un message de diffusion à un saut. Il contient les champs suivants :
    - l'identifiant du nœud émetteur,
    - le niveau d'énergie restant du nœud émetteur,
    - le compteur des messages (R),
    - l'identifiant du nœud suspect,
    - la valeur actuelle du compteur L correspondant au nœud suspect,
    - le niveau d'énergie restant correspondant au nœud suspect.

4. Message d'acquittement (A) : message envoyé par un nœud pour acquitter la réception du message du rejet concernant un nœud suspect. A la réception du message d'acquittement, le nœud récepteur arrête d'envoyer des messages de rejet (R) concernant le nœud suspect. La taille du message (A) est de 10 octets. Ce message est un message de type « unicast » à un saut. Il contient les champs suivants :
    - l'identifiant du nœud émetteur,
    - le niveau d'énergie restant,

- le compteur de messages (A),
- l'identifiant du nœud suspect.

Le service utilise aussi les messages périodiques des autres protocoles, par exemple le synchronisation, ainsi que les paquets des données pour mettre à jour l'état des nœuds voisins. Pour minimiser la consommation d'énergie, les messages sont envoyés à un seul saut (Time To Live=1) de la portée de la communication.

Emetteur (2 octets)	Energie (2 octets)	Compteur (4 octets)

(a)

Emetteur (2 octets)	Energie (2 octets)	Compteur (4 octets)	Suspect (2 octets)

(b)

Emetteur (2 octets)	Energie (2 octets)	Compteur (4 octets)	Suspect (2 octets)	L (1 octet)	Energie (2)

(c)

Emetteur (2 octets)	Energie (2 octets)	Compteur (4 octets)

(d)

Figure 4-6 : Format des messages : (a) Hearbeat, (b) Requête, (c) Rejet, (d) Acquittement

4.3.4. L'algorithme de détection

Comme nous l'avons indiqué précédemment, la détection des éléments défaillants s'effectue en deux étapes : l'étape de *la détection locale* et l'étape du *consensus (*Figure 4-7*)*.

- *A l'état initial,* chaque nœud crée la table de voisinage en fonction du seuil de la qualité du lien (*LQmin*), du nombre minimal (*Nmin*) et du nombre maximal (*Nmax*) des voisins configurés par la couche d'application.

  Une fois la table créée, la phase de *détection locale* commence. Pendant cette phase, le compteur *L* se décrémente pour chaque période active durant laquelle un nœud surveillant (i) n'a pas reçu un message d'un nœud voisin spécifique, par exemple un nœud (j). A la réception de n'importe quel type de message, le

compteur *L* est réinitialisé. Si le compteur L correspondant au nœud j arrive à zéro, le nœud i considère le nœud j comme un nœud suspect et la phase de consensus commence.

- Pendant la *phase de consensus*, le nœud surveillant (i) diffuse un message de requête (*Q*) pour vérifier l'état du nœud (j) et le compteur *C* se décrémente à chaque période active, pendant laquelle le nœud surveillant (i), n'a pas reçu un message concernant le nœud suspect (j). Pendant cette phase, si le nœud i reçoit un message de rejet (**R**) d'un nœud, par exemple k correspondant au nœud j, il le considère en bon état (retour à la *phase de la détection locale*). Dans ce cas là, le nœud i envoie un message d'acquittement (***A***) au nœud k pour confirmer le rejet. Dans le cas où le compteur *C* arrive à zéro, le nœud i prend une décision finale concernant la défaillance du nœud j.

  Ensuite, la couche application, sera notifiée de la présence d'un nœud isolé à la fin de la période de notification. Une fois la notification envoyée, le nœud j est supprimé de la table des voisins du nœud i. Notons qu'un nœud s'implique dans des étapes différentes pour chaque voisin séparément.

4.3.5. L'algorithme de diagnostic

Les causes principales des pannes notifiées par le service SMART sont :

- l'épuisement de la batterie,
- la défaillance du lien,
- une cause inconnue.

Le service différencie les causes des défaillances (le non réception des messages) en se basant sur l'information du niveau d'énergie et la qualité du lien correspondant à chaque nœud. L'information du niveau d'énergie est incluse dans chaque paquet envoyé par un nœud (marquage des paquets). La qualité du lien séparant deux nœuds est mesurée par un nœud à chaque réception d'un paquet de la communication. Si le niveau d'énergie est inférieur à un certain seuil, le service de diagnostic notifie qu'il y a un problème d'épuisement d'énergie. Similairement, si la qualité du lien est inférieure à un certain seuil, le service notifie qu'il y a un problème de lien (Figure 4-8).

4.3.6. Interfaces du service

Le service fournit plusieurs interfaces à la couche utilisatrice.

1. La commande de *configuration* (« Configure ») permet de déterminer et éventuellement d'ajuster les valeurs des paramètres du service :
   - le compteur qui correspond à la phase de la détection locale *L*,
   - le compteur qui correspond à la phase du consensus *C*,
   - le nombre minimal de voisins *Nmin*,

- le nombre maximal de voisins *Nmax*,
- le seuil minimal du niveau d'énergie (*Emin*),
- le seuil minimal de la qualité du lien (*LQmin*),
- les temporisateurs de plan du cycle d'activité du nœud,
- la période de la notification des pannes.

2. Les commandes de *lancement et d'arrêt* du service (« Start/Stop »). Elles permettent à la couche utilisatrice de contrôler l'activité du diagnostic en fonction de l'exigence de la qualité de service requise par l'application. Par exemple, l'administrateur peut arrêter le service de diagnostic, au début du déploiement du réseau, pour minimiser la consommation d'énergie. Il peut aussi choisir de lancer le service, dans les zones les plus prioritaires à diagnostiquer, comme dans des zones où il y a un niveau de redondance faible, et d'arrêter le service dans les zones fiables après avoir analysé des données historiques.

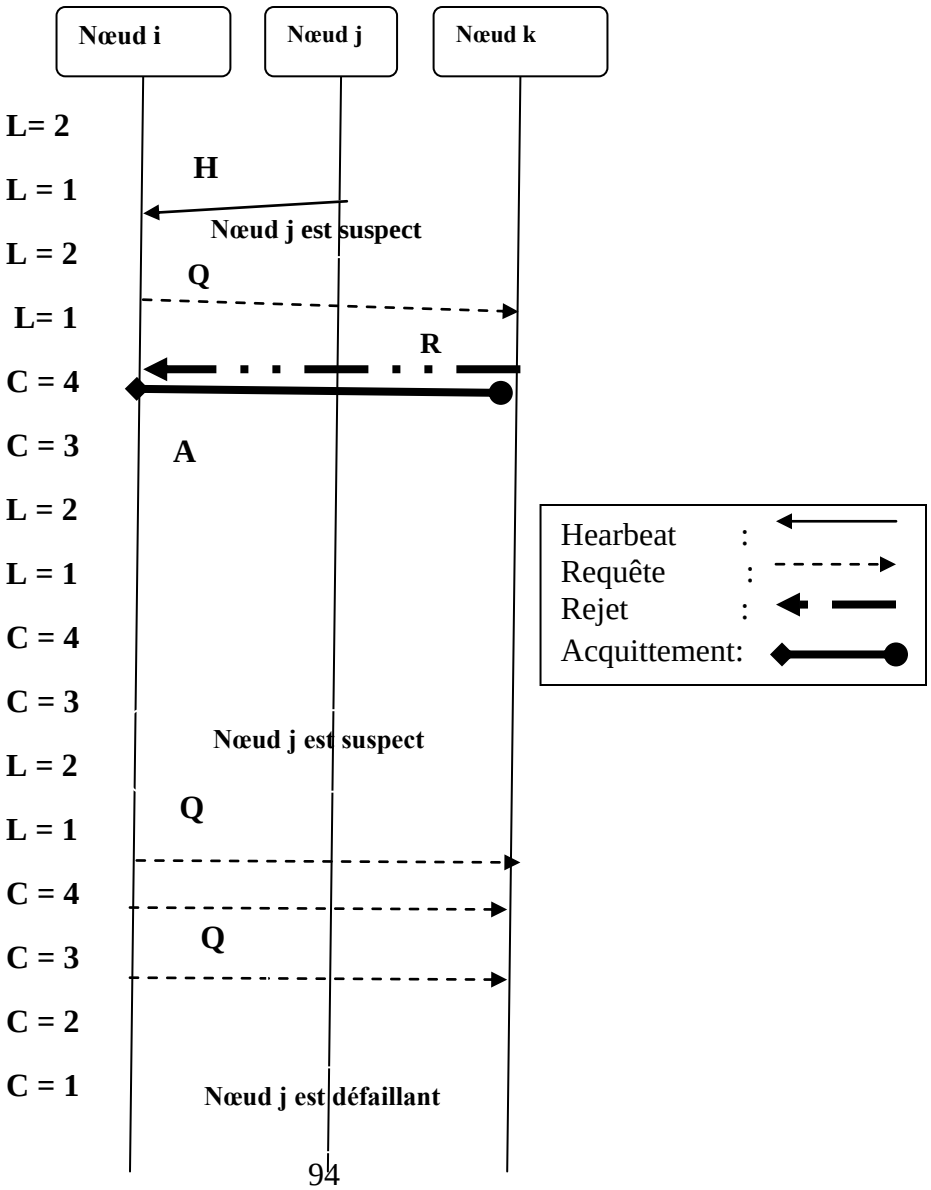


Figure 4-7 : Processus de détection

3. La commande de *notification* des pannes (« Report »). Le service fournit deux types de rapports de pannes :
   - Rapport des pannes détectées pendant la phase de détection locale.
   - Rapport des pannes détectées pendant la phase du consensus.

L'application peut choisir un ou deux types de rapports, ainsi que la période de notification des pannes en utilisant la commande de la configuration. En effet, le premier type de rapport peut être important pour certaines applications, qui ont besoin d'informations sur les pannes temporaires des liens. Le rapport inclut les informations suivantes :

- l'identifiant du nœud défaillant,
- le type de rapport,
- la sévérité des pannes (la sévérité dépend des valeurs courantes des compteurs *L* et *C*.)
- la cause estimée de la défaillance.

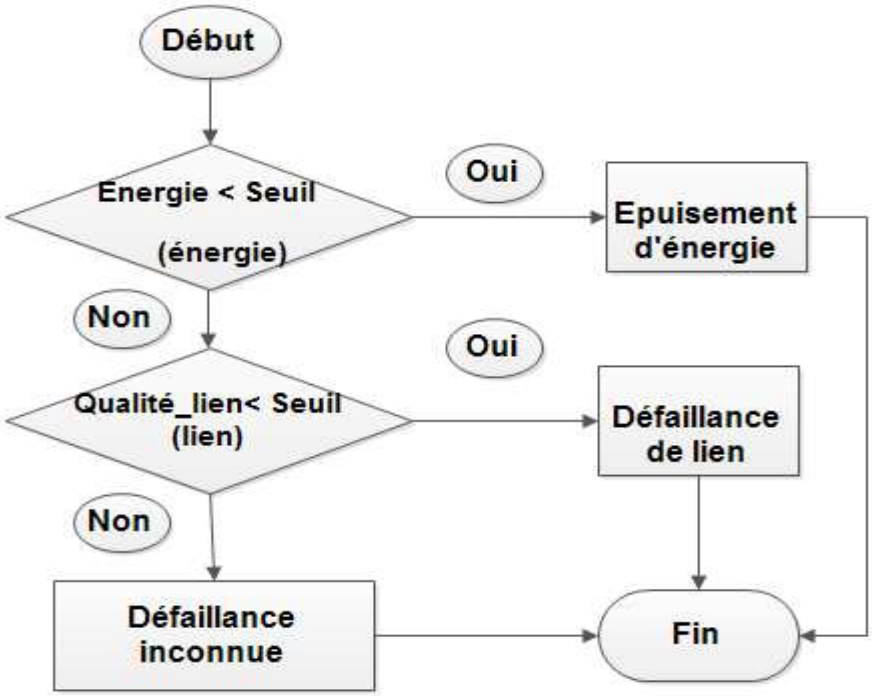


Figure 4-8 : Algorithme de diagnostic des fautes

4.4.Simulation

La simulation d'un réseau de capteurs, avant son déploiement réel, représente une alternative moins coûteuse et moins lourde que son expérimentation en environnement réel. Actuellement, il y a de nombreuses plateformes de simulation de réseaux de capteurs disponibles sur le marché, qu'elles soient commerciales ou non, parmi

lesquelles : Ns-2 [96], OMNET++ [95], Tossim[93], SensorSim [153], UWSim[154], Avrora[155]. Etant donnée cette variété de simulateurs, nous avons choisi celui qui répond au mieux aux besoins de nos simulations. Notre choix s'est porté sur le simulateur TOSSIM [93] qui fonctionne avec le système d'exploitation TinyOS [21].

4.4.1. Simulateur

TinyOS est un système d'exploitation intégré, modulaire, libre et open source destiné aux RCSF. TinyOS s'appuie sur le langage NesC [156]. Le langage NesC, est comme son nom l'indique à base de langage C, à la différence qu'il intègre la notion de composants, afin de rendre un programme modulaire et réutilisable. Celui-ci propose une architecture permettant de réduire considérablement la taille mémoire du système, et de ses applications. Chaque composant correspond à un élément matériel (LEDs, timer, ADC…) et peut être réutilisé dans différentes applications. Les composants peuvent être des concepts abstraits ou bien des interfaces logicielles aux entrées-sorties matérielles de la cible étudiée (Figure 4-9). Ce qui nous motive à adopter TinyOS est son acceptation dans la communauté RCSF et sa portabilité sur de nombreuses plateformes matérielles. En outre, la structure à base de composants favorise l'architecture logicielle de notre service. Ensuite, la performance de notre solution est facilement étudiée en utilisant TOSSIM.

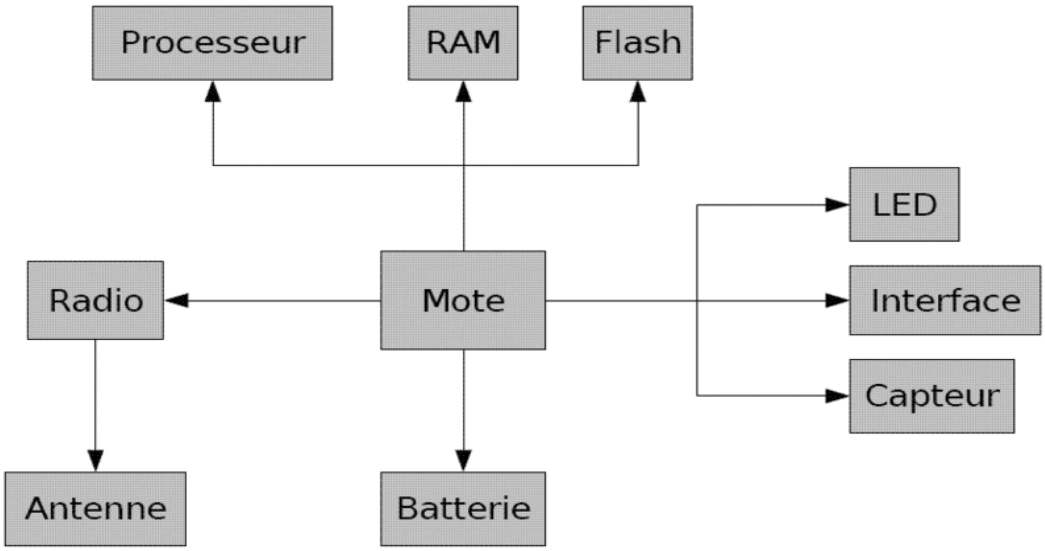


Figure 4-9: Architecture générale des cibles utilisant TinyOS [157]

TOSSIM simule les couches physiques et la couche liaison de données. Il permet de simuler le comportement d'un capteur (envoi/réception de messages via les ondes radios, traitement de l'information, ...) au sein d'un réseau de capteurs. TOSSIM permet la simulation des applications TinyOS en remplaçant des composants bas niveau par l'implémentation de simulations. L'application TinyOS et les modules de TOSSIM sont compilés et liés à travers une librairie. Cette librairie ainsi qu'un interpréteur Python, sont utilisés pour définir la topologie du réseau, pour configurer et lancer la simulation.

Python permet l'interaction dynamique avec la simulation ainsi que le changement (inspection) de variable dans un programme TinyOS lancé.
Durant notre phase de simulation, nous avons exécuté plusieurs scénarii, en variant à chaque fois le type de déploiement, le nombre de nœuds, le temps d'exécution ou le nombre de pannes dans le réseau, *etc*. Ceci permet d'analyser chaque paramètre d'évaluation afin d'étudier l'impact de la solution proposée.

4.4.2. Modèle réseau

Le déploiement de capteurs est un élément de conception essentiel pour de nombreuses applications des RCSF. L'objectif principal est d'assurer une zone de couverture complète pour le champ d'observation d'un phénomène physique. En fait, trois types de déploiements ont été proposés dans la littérature[158] :

- déploiement *déterministe* spécifique à l'application : les nœuds de capteurs sont placés délibérément dans la zone souhaitée. Ce type de déploiement est approprié seulement pour une application à petite échelle.
- déploiement *aléatoire* : les nœuds sont positionnés aléatoirement pour former un RCSF. Ce type de déploiement permet le passage à l'échelle des applications, surtout dans un cas de déploiement en environnement hostile. Cependant, il pourrait être coûteux, puisqu'il nécessite un certain niveau de redondance, pour garantir la robustesse du réseau.
- déploiement à *base de grille* : connu également sous le nom de déploiement à base de modèle. L'approche de déploiement à base de grille est une approche intéressante pour les déploiements à moyenne et à grande échelle en raison de sa simplicité et de son évolutivité.

Nous avons étudié les performances de la solution en nous basant sur deux types de stratégies de déploiement : déploiement à base de grille (Figure 4-10) et déploiement aléatoire (Figure 4-11). Lorsqu'on se réfère à une grille de taille x, on désigne une grille contenant les x nœuds chacun à 10m. Dans les différentes densités/topologies montrées dans la simulation, les nœuds sont placés dans une région de 40m x 40m en fonction de la topologie choisie.

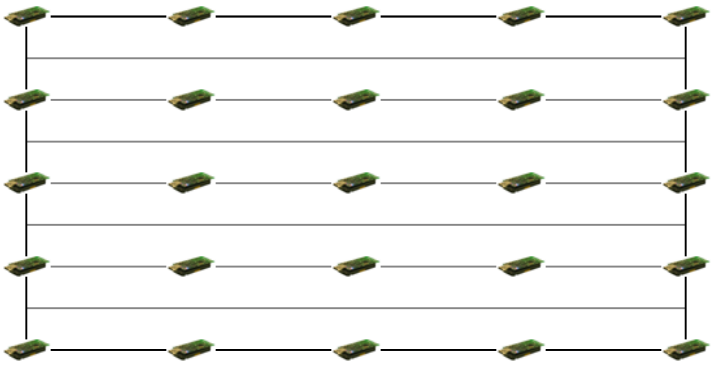


Figure 4-10: Déploiement à base de modèle (grille) d'un RCSF

4.4.3. Modèles de défaillances

Nous avons étudié les performances de notre service suivant deux modèles de défaillance :

- Défaillances *isolées* : ce modèle représente des défaillances indépendantes des nœuds. Il représente souvent les effets de l'environnement local. Pour simuler les défaillances isolées, on a déconnecté des nœuds choisis au hasard et à des moments aléatoires en se basant sur le processus de Bernoulli. La déconnexion d'un nœud est simulée par la suppression de tous ses liens dans le modèle de radio TOSSIM. Les nœuds déconnectés, ont été reconnectés après un intervalle de temps spécifique. De plus, les nœuds qui ont atteint le niveau d'énergie minimal ont été interrompus, afin de simuler la défaillance des nœuds, due à l'épuisement de la batterie.
- Défaillances *à base de modèles* : ce modèle représente des défaillances de nœuds corrélés géographiquement, c'est-à-dire la défaillance des nœuds qui se trouvent dans une même zone géographique. Pour simuler les défaillances à base de modèles, un centre et un rayon d'une zone défectueuse ont été spécifiés en premier lieu, puis tous les nœuds dans cette zone ont été déconnectés.

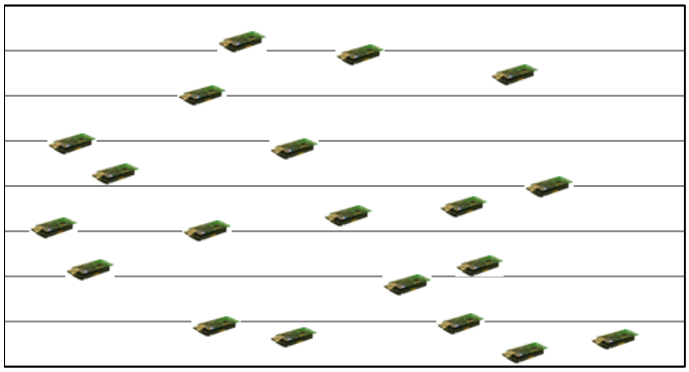


Figure 4-11: Déploiement aléatoire d'un RCSF

4.4.4. Modèle de la consommation d'énergie

Comme les études expérimentales ont montré que, généralement, l'énergie de communication représente la portion la plus grande de l'énergie consommée par un nœud capteur [159], nous avons intégré dans notre étude le coût du module radio pour évaluer la consommation d'énergie de notre approche. Nous avons calculé la consommation d'énergie selon l'approche présentée par Plastre et. *al* [160]. Celle-ci se base principalement sur les périodes de transmission de données, les périodes de réception des données et les périodes oisives.

Considérons que voltage, courant (tx/rx), temps (tx/rx) désignent respectivement le voltage, le courant consommé en transmission/réception et le temps passé à transmettre/à recevoir. La consommation d'énergie nécessaire à la transmission/ à la réception d'une trame de données (Energie (tx/rx)) sera calculée par la formule suivante :

$$\text{Energie (tx/rx)} = \text{Voltage} * \text{Courant (tx/ rx)} * \text{temps (tx/rx)} \quad (4.1)$$

où le courant (tx/rx) est exprimé en Ampères, le voltage en Volts, le temps en secondes et l'énergie en Joules. Les valeurs de Voltage et Courant (tx/rx) sont des caractéristiques de la plateforme utilisée. Les caractéristiques de la plateforme « Micaz» utilisée dans la simulation peuvent être trouvées dans les feuilles de données de la plateforme (Tableau 4-1). Le temps passé à transmettre/à recevoir est calculé par la formule suivante :

$$\text{temps(tx/rx)} = \frac{taille\ dela\ trame\ des\ données}{debit\ du\ canal} \quad (4.2)$$

où la taille des données est exprimée en bits et le débit du canal en kbps.
La même formule que (4.1) est utilisée pour calculer l'énergie consommée pendant les périodes oisives (Energie (oisive)). Considérons voltage, courant (oisive), temps (oisive) désignent respectivement le voltage, le courant consommé en mode oisif et la période d'oisiveté.
La consommation d'énergie pendant la période oisive sera calculée par la formule suivante :

$$\text{Energie (oisive)} = \text{Voltage} * \text{Courant (oisive)} * \text{temps (oisive)} \quad (4.3)$$

Ainsi, la consommation totale d'énergie à la nième période d'activité « duty- cycle » T(n) sera calculée par la formule suivante :

$$T(n) = \sum_{1}^{n} (Energie(tx) + Energie(rx) + Energie(oisive)) \quad (4.4)$$

Finalement, le niveau d'énergie restant R(n) à la nième « duty- cycle » sera calculé par la formule suivante :

$$R(n) = Energie(0) - T(n) \quad (4.5)$$

Où Energie (0) représente la réserve initiale d'énergie du capteur. La réserve initiale peut être déterminée par la feuille des données spécifique à la plateforme utilisée.

Débit du canal	250 kbps
RAM	4 Kbytes
ROM	128 Kbyes
Radio en transmission	17.4 mA
Radio en réception	19.7 mA
Radio en état oisif	20 µA
Voltage	2.7 V -3.3 V
Batterie	2xAA batteries

Tableau 4-1: Caractéristiques de la plateforme « Micaz »

4.4.5. Détermination des paramètres

Nous avons étudié la performance de la solution en déterminant les valeurs des paramètres de service de la façon suivante (Tableau 4-2).

- *Paramètres de la table de voisinage* : nous avons déterminé le nombre minimal de voisins (*Nmin*), le nombre maximal de voisins (*Nmax*) et le seuil minimal de la qualité de lien (*LQmin)* comme étant : 4, 6, -90 db respectivement. Ces valeurs permettent d'éviter la présence de nœuds isolés et garantissent un niveau de robustesse tout en limitant la consommation d'énergie nécessaire à surveiller un ensemble déterminé de nœuds voisins.
- *Temporisateurs* : Nous avons fixé la période de détection et la période d'activité des nœuds du réseau à 1000 ms et 100 ms respectivement. La période d'activité est liée au temps de latence maximal de la couche MAC (10 ms dans notre cas). Une période de 1000 ms de détection permet alors une période de sommeil de 900 ms. La période de notification est fixée pour être égale à la période de la détection. De cette façon, la notification des pannes s'effectue dès qu'elles se produisent.
- *Compteurs :* Les compteurs de la phase de détection locale (*L*) et la phase de consensus (*C*) sont fixés à 4 et 5 (nombre supérieurs à 1), ceci afin de réduire le taux de fausses alertes en cas de perte d'un seul message. De cette façon, c'est uniquement lorsqu'il y a une perte de 4 messages d'un nœud voisin qu'on le considère comme étant un nœud suspect. De la même façon, il faut avoir 5 messages pour juger définitivement de l'état d'un nœud voisin.

Paramètre	Valeur
Nmin	4 nœuds
Nmax	6 nœuds
LQmin	-90 db
Période de détection	1000 ms
Période d'activité	100 ms
Période de sommeil	900 ms
Période de notification	1000 ms
L	4 messages
C	5 messages

Tableau 4-2: Paramètres de la simulation

Tous les paramètres évoqués peuvent être reconfigurés en ligne pour s'adapter à la dynamique de déploiement. Le processus de configuration est effectué en envoyant un message spécial aux nœuds du réseau contenant les nouvelles valeurs des paramètres évoqués ci-dessus.

4.5.Evaluation de la performance

Nous avons utilisé plusieurs critères afin d'évaluer les performances du service de diagnostic :

- *les coûts* (mémoire et énergie),
- *L'exactitude de détection* des défaillances (défaillances isolées et à base de modèles pour les déploiements de type aléatoire et à base de grilles, pour différentes tailles du réseau, pour différentes durées de défaillances). L'exactitude de détection est calculée par la formule suivante :

$$Exactitude\ de\ \text{détection} = \frac{Nombre\ des\ nœuds\ détectés\ comme\ défectueux}{Nombre\ des\ nœuds\ défectueux} \quad (4.6)$$

- *L'exactitude de diagnostic* pour différentes cause de défaillances (épuisement de la batterie, mauvaise connectivité de lien ou cause imprévue) et pour différentes tailles du réseau. L'exactitude de diagnostic d'une defaillance due à une cause X est calculée par la formule suivante :

$$Exactitude\ de\ \text{diagnostic} = \frac{Nombre\ des\ défaillances\ dues\ à\ la\ cause\ X\ diagnostiquées\ correctement}{Nombre\ des\ défaillances\ dues\ à\ la\ cause\ X} \quad (4.7)$$

Dans la suite, nous présenterons les résultats de la simulation qui se stabilisent en répétant l'expérimentation cinq fois.

4.5.1. Coût mémoire

Le service de diagnostic réside dans chaque nœud du réseau. Le coût mémoire nécessaire pour l'implémenter sur la plateforme Micaz TinyOS est d'environ 14 Kbytes en ROM et 1 Kbytes en RAM. La taille mémoire a été contrôlée par la politique de gestion des voisins. Cette politique limite le nombre de voisins surveillés par chaque nœud. Par conséquent, elle réduit la taille de la table de voisinage maintenue par un nœud.

4.5.2. Coût énergétique

La consommation d'énergie est une métrique importante influant sur la qualité de service du RCSF. Les résultats expérimentaux montrent que le service passe à l'échelle :

- La moyenne de la consommation d'énergie s'élève à 200 (mj) pour une durée de fonctionnement de 8 minutes. La valeur maximale s'élève à (400 mj). La consommation s'élève légèrement avec l'augmentation de la taille du réseau (25, 39, 64, 81 et 100) et varie peu avec le type de déploiement (aléatoire ou grille) (Figure 4-12).
- La consommation d'énergie augmente avec la diminution de la durée de la période de détection, (Figure 4-13). Cette diminution permet de réduire la latence de détection (Figure 4-14). Celle-ci est calculée par la formule suivante :

$$\text{Latence de détection } = (L + C) * N \qquad (4.6)$$

Où, L, C et N désignent respectivement le compteur de la phase de détection locale, le compteur de la phase de consensus et le numéro du cycle d'activité du nœud.

En effet, le service maîtrise la consommation d'énergie de plusieurs façons :

- La *communication à un saut*. La diffusion des messages est limitée à un saut (TTL=1). Cela permet d'éviter le coût élevé de la communication multi-saut.
- Le mécanisme de *marquage* qui consiste à inclure l'information de diagnostic, (ici, le niveau d'énergie restant), dans chaque message transmis par un nœud. Donc, les nœuds, dans le service présenté, n'ont pas besoin de transmettre des messages spécifiques pour le diagnostic dans le réseau. En outre, les messages des protocoles et les messages des données sont exploitées pour détecter les défaillances.

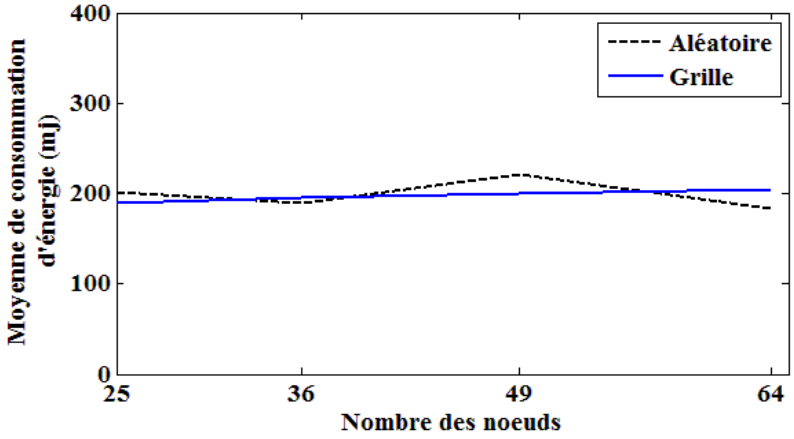


Figure 4-12 : Moyenne de consommation d'energie

- La politique de *gestion des voisins*. Le but de cette politique est de limiter le nombre des voisins surveillé par un nœud. Elle permet alors de limiter l'échange des messages tout en respectant les besoins de l'application en termes de qualité de service.
- La politique de *gestion des temporisateurs de service* (périodes de cycle d'activité, période de détection, période de notification, *etc*.). Ainsi, une politique stricte ou tolérante des temporisateurs affecte considérablement la consommation d'énergie. Normalement, les temporisateurs sont déterminés en se basant sur des études et des calculs effectués avant le déploiement. Ces temporisateurs restent fixes en dépit des changements pendant la durée du déploiement et du scenario de déploiement. Les temporisateurs fixes peuvent gaspiller les ressources radio puisque les nœuds sont configurés afin de surveiller périodiquement (et continuellement) leur voisinage, même dans les surfaces qui semblent être sécurisées, denses et stables du réseau.

  La politique ici considère le *scenario de déploiement* (environnement intérieur ou extérieur, réseau dense ou dispersé, zones risquées ou sécurisées, applications critiques ou non) ainsi que sa *dynamique* dans le temps. Ainsi, la période de détection peut être raccourcie (ou alternativement, le cycle d'activité peut être allongé) dans le cas des applications critiques, afin d'améliorer les aspects réels de diagnostic. Par contre, la période de détection

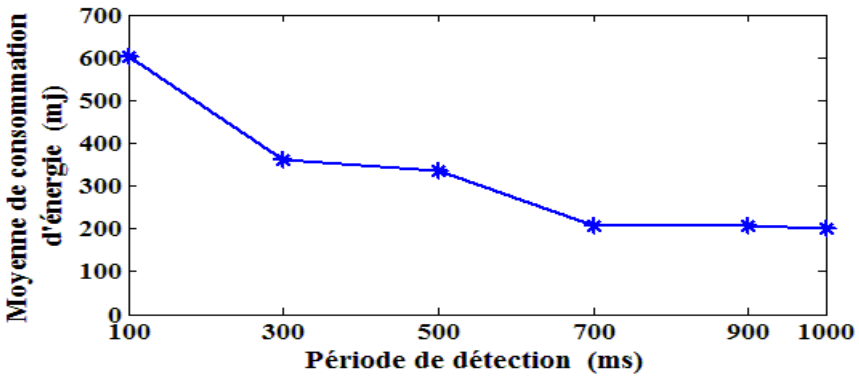


Figure 4-13: Impact de la période de détection sur la consommation d'énergie

peut être rallongée dans de nombreux scenarii, par exemple, dans les cas d'applications non critiques, de déploiements denses, de déploiements intérieurs et stables (le taux des défaillances des liens sera éventuellement bas), ainsi que dans tous les scenarii qui semblent être sécurisés. De même, le changement de la politique au cours de la vie du réseau pourra être bénéfique en termes d'énergie. Par exemple, au début du déploiement, la puissance des batteries est généralement bonne (le taux des défaillances des nœuds à cause de l'épuisement d'énergie sera éventuellement bas). Par conséquent, l'application d'une politique tolérante, au début du déploiement, puis une politique plus stricte au cours du temps permettra de rationaliser la consommation d'énergie.

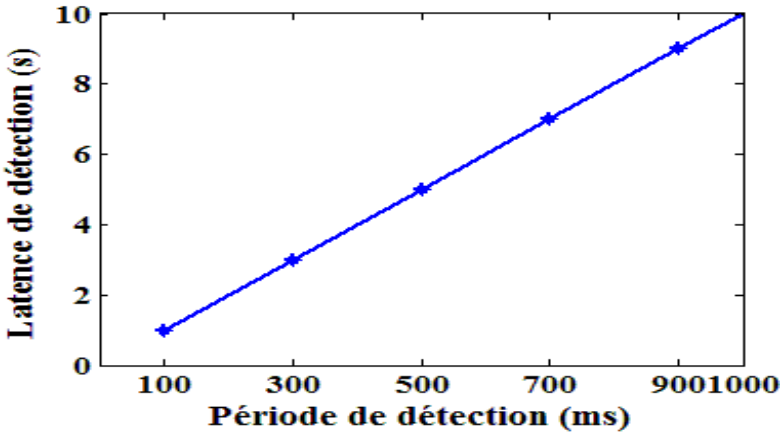
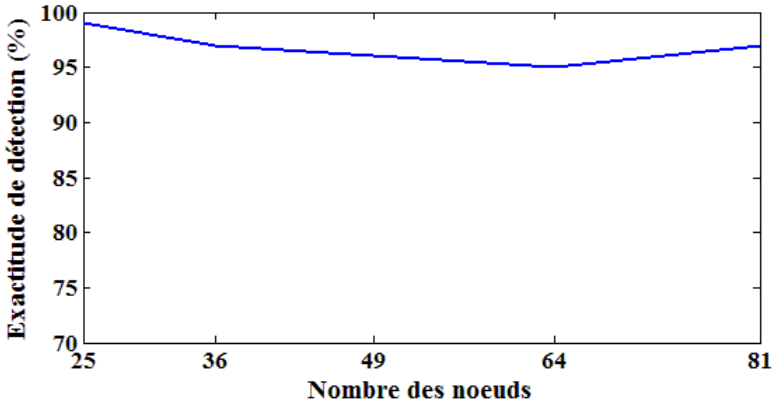


Figure 4-14: Impact de la durée de la période de détéction sur la latence de détection

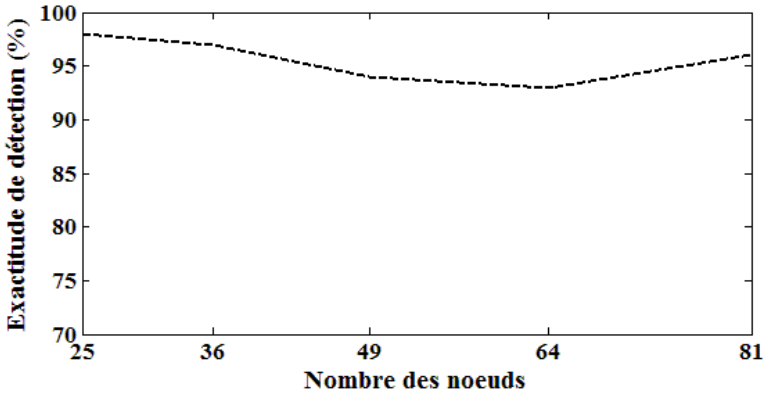
4.5.3. Exactitude de la détection

L'exactitude de la détection mesure l'aptitude du système de diagnostic à pouvoir déceler la présence d'une défaillance. Elle est fortement liée à la notion de non-détection. Nous avons étudié le taux de détection ou l'exactitude de détection en variant plusieurs paramètres. Dans tous les cas, les résultats expérimentaux montrent l'efficacité du service en terme de détection :

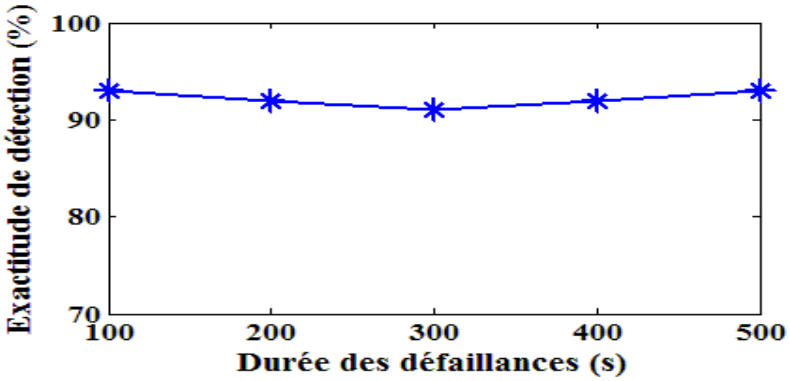
- *Défaillances isolées* : La moyenne de détection des défaillances isolées est toujours supérieure à 90% pour différentes tailles de réseaux, différents types de déploiement et différentes durées de défaillances (Figure 4-15).
- *Défaillances à base de modèle* : La Figure 4-16 montre l'exactitude de la détection en présence des défaillances à base de modèles où l'exactitude de la détection diminue avec l'augmentation du rayon. Si la surface de la zone défectueuse est petite (R<20 m), le service détecte les défaillances, avec un taux satisfaisant supérieur à 60%. Par contre, si la surface de la zone défectueuse est grande (R> 20), le service détecte les défaillances avec un taux inférieur à 50%. C'est parce que les nœuds peuvent surveiller uniquement les nœuds au niveau du périmètre de la zone défectueuse, et non plus ceux qui se trouvent à l'intérieur de la zone.



(a) Déploiement à base de modèle (grille)



(b) Déploiement de type aléatoire



(c) Défaillances de durée différente

Figure 4-15 : Exactitude de la détection des défaillances

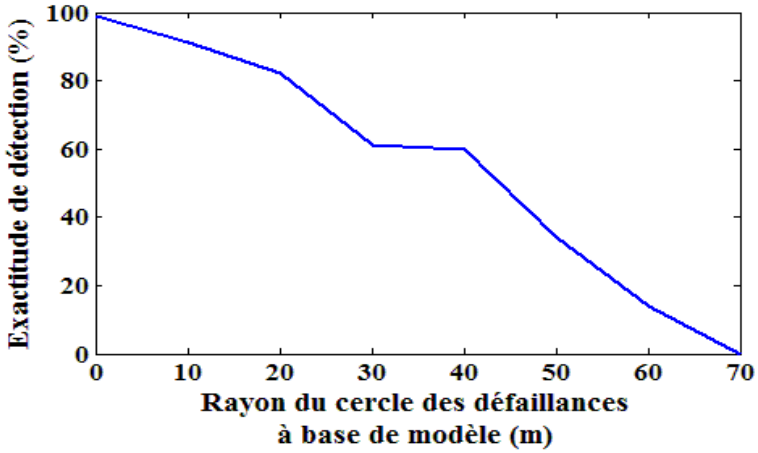


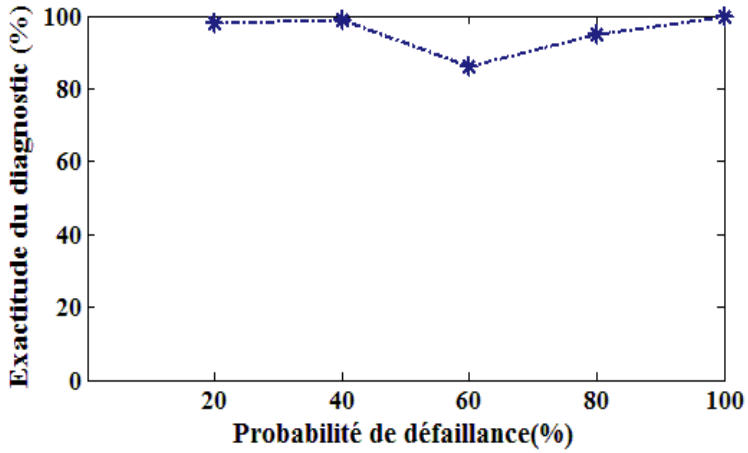
Figure 4-16: Exactitude de la détection des défaillances à base de modèles

4.5.4. Exactitude du diagnostic

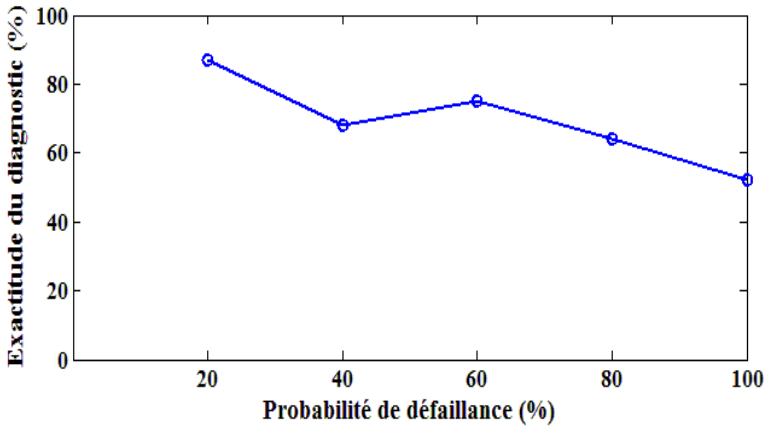
L'exactitude du diagnostic mesure la capacité de remonter directement à l'origine du défaut. Une défaillance engendre souvent une cascade d'alarmes et il peut être difficile de remonter à l'élément défaillant. La Figure 4-17 montre que le service fournit des résultats satisfaisants où le taux d'exactitude est égal à :

- 95% en moyenne pour la défaillance résultante de l'épuisement de l'énergie,
- 70% en moyenne pour la défaillance résultante des problèmes du lien,
- 30% en moyenne pour une défaillance imprévue.

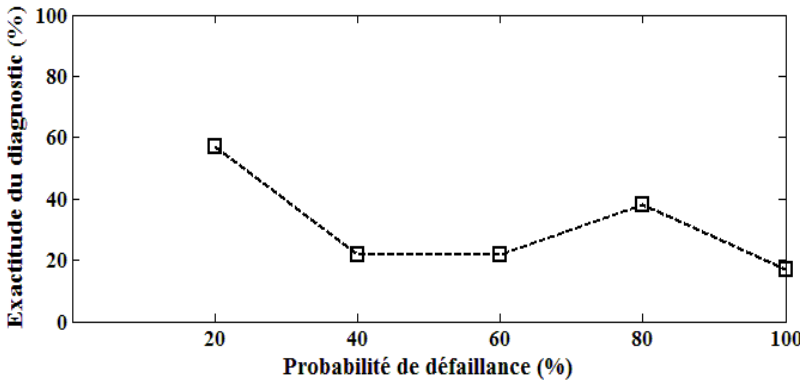
Le diagnostic de la cause principale de la défaillance imprévue est difficile, malgré un taux de détection de pannes assez élevé.



(a) Défaillance des nœuds à cause de l'épuisement de la batterie



(b) : Défaillance des liens



(c)Défaillance imprévue

Figure 4-17: Exactitude du diagnostic

4.6. Conclusion

Le but de ce chapitre est de présenter un aspect important de notre travail de thèse. Il s'agit d'un service de diagnostic adopté dans notre approche globale pour la tolérance aux fautes dans le RCSF. Le service, représente également une approche autonome du diagnostic des fautes, adaptée aux systèmes de RCSF. Il offre les services de diagnostic, indépendamment du type d'application, du type de déploiement et des protocoles des couches MAC et de routage. Du point de vue conception, le service est implémenté dans une couche indépendante de la pile protocolaire, mais facilement intégré avec les couches de la pile. Il fournit aux applications de nombreux paramètres ajustables qui le rendent approprié pour de nombreux types de déploiement. En ajustant les paramètres, l'application peut réaliser le compromis requis entre les différents enjeux de la qualité de service : la latence de détection, la consommation énergétique, la robustesse, la perte de paquets tolérable afin de réduire le taux de fausses alertes, *etc.* Le service est *adaptatif* vis-vis des changements topologiques du réseau. Il s'adapte à l'ajout, au retrait des nœuds et aux cassures des liens en gérant dynamiquement le nombre optimal des voisins à surveiller. Enfin, les résultats expérimentaux montrent l'efficacité du service en termes de consommation des ressources, tout en fournissant un taux de détection et une précision de diagnostic satisfaisants.

Chapitre 5.

IFTF : APPROCHE GLOBALE DE TOLERANCE AUX FAUTES DANS les RCSF

Ce qui importe n'est pas tant la fiabilité des capteurs du système que la fiabilité du système dans son ensemble. Est-ce que le système surveille correctement la pollution de l'air ? Est-ce que le système mesure correctement les conditions météorologiques ? Est-ce que le système détecte correctement les feux et dans les meilleurs délais ? Dans les deux chapitres précédents, nous avons présenté deux approches pour la gestion des fautes à trois niveaux : données, nœuds et réseaux. En se basant sur les résultats de ces approches, nous présentons dans ce chapitre, une nouvelle approche de la tolérance aux fautes, nommée « IFTF ». Les caractéristiques majeures d'IFTF sont qu'elle est une approche globale, générale et adaptable aux systèmes à base de réseaux de capteurs sans fils. IFTF se base sur le test des services d'application avec la possibilité de diagnostiquer les causes des défaillances et d'isoler les zones défectueuses tout en rationnalisant la consommation d'énergie.

SOMMAIRE

5.1. Introduction

La tolérance aux pannes est un défi crucial dans les RCSF en raison de divers types de fautes qui peuvent être produites. Toutefois, une question importante se pose :

Quelles sont les éléments ou les états d'incohérences du système qui doivent être examinés par les outils de diagnostic et de débogage?

Un diagnostic global de tous les éléments du système est impossible à cause de la forte consommation d'énergie qui serait nécessaire. C'est pourquoi, les outils de diagnostic traditionnels [55][61][70] sélectionnent un ensemble déterminé de type de fautes, afin d'améliorer la fiabilité du système. Cependant, cela ne suffit pas pour répondre aux quatre défis majeurs concernant la fiabilité du système en ligne :

1) L'évaluation positive de l'outil de diagnostic ne garantit pas la sûreté de fonctionnement du système au niveau de la *sémantique de l'application* ;
2) L'évaluation négative de l'outil de diagnostic ne signifie pas forcément qu'il y a un problème au niveau de l'application, à cause de la nature de tolérance des fautes inhérente aux systèmes de RCSF, en raison notamment de la redondance ;
3) La dynamique de l'environnement, la reconfiguration, la reprogrammation et toute évolution dans les conditions du déploiement peuvent conduire à un dysfonctionnement imprévu pendant le fonctionnement du système, ce qui finalement peut compromettre la fiabilité du système ;
4) Les nœuds de capteurs peuvent se désynchroniser et les applications peuvent facilement atteindre des états arbitraires en raison d'une configuration inadéquate du système conduisant ainsi à l'effondrement en ligne de certaines hypothèses considérées à la conception.

Pour illustrer la nécessité d'une approche globale de tolérance aux fautes, nous présenterons un scenario d'application pour la prédiction d'explosion dans une mine de charbon, ayant toutes les contraintes des applications de sécurité critiques des RCSF.

5.2. Application de RCSF pour la prédiction d'explosion

Considérons une mine de charbon souterraine qui a une quantité négligeable d'oxygène, mais pouvant générer beaucoup de gaz inflammables (méthane, CO_2, *etc*.) [161]. La mine est grande et possède certains trous au niveau de la surface du sol permettant la pénétration des rayons solaires à l'intérieur de la mine. Le gouvernement local porte un certain intérêt et une certaine inquiétude quant à cette mine, suite à des explosions qui se produisent de temps en temps, nuisant ainsi aux villages voisins. L'inspection dévoile que la cause majeure de l'occurrence de l'explosion provient du processus de photosynthèse produisant l'oxygène. Ce processus résulte d'une croissance de plantes vertes à l'intérieur de la mine (environnement hostile), et à la pénétration des rayons solaires à travers les trous du sol. Le mélange en quantité suffisante d'oxygène avec le méthane accompagné

d'une petite étincelle de feu (une étincelle naturelle est probable si une roche tombe et frappe une autre roche), peut entraîner une explosion significative.
Dans un tel scenario, il est impossible de recourir aux interventions humaines, pour surveiller et contrôler un environnement hostile et dangereux. D'où l'importance, de mettre en œuvre une application de réseau de capteurs sans fils afin de surveiller la mine, et envoyer en temps réel, des informations critiques (des alertes) avant qu'une explosion se produise. Une telle application ne doit pas utiliser des capteurs qui détectent directement la présence d'oxygène, car dans ce cas il peut être déjà trop tard. Il est impératif de détecter le processus de photosynthèse afin de prendre toutes les précautions nécessaires permettant de diminuer le danger. Dans cette application, six types de capteurs sont nécessaires pour prédire l'explosion (Figure 5-1) :

1) Un capteur biologique (S1) pour détecter la présence de mousse verte ;
2) Un capteur de lumière (S2) pour détecter les rayons solaires ;
3) Un capteur chimique (S3) pour détecter le gaz CO_2 ;
4) Un capteur chimique (S4) pour détecter un taux élevé de méthane ;
5) Un capteur de température (S5) pour détecter la génération de la chaleur ;
6) Un capteur sensible à la lumière (S6) pour détecter les étincelles de lumière.

La photosynthèse est détectée par les capteurs de mousse verte (S1), de lumière (S2) et de gaz CO_2 (S3). Les étincelles de feu sont détectées par les capteurs de température (S5) et le capteur des étincelles de lumière (S6).

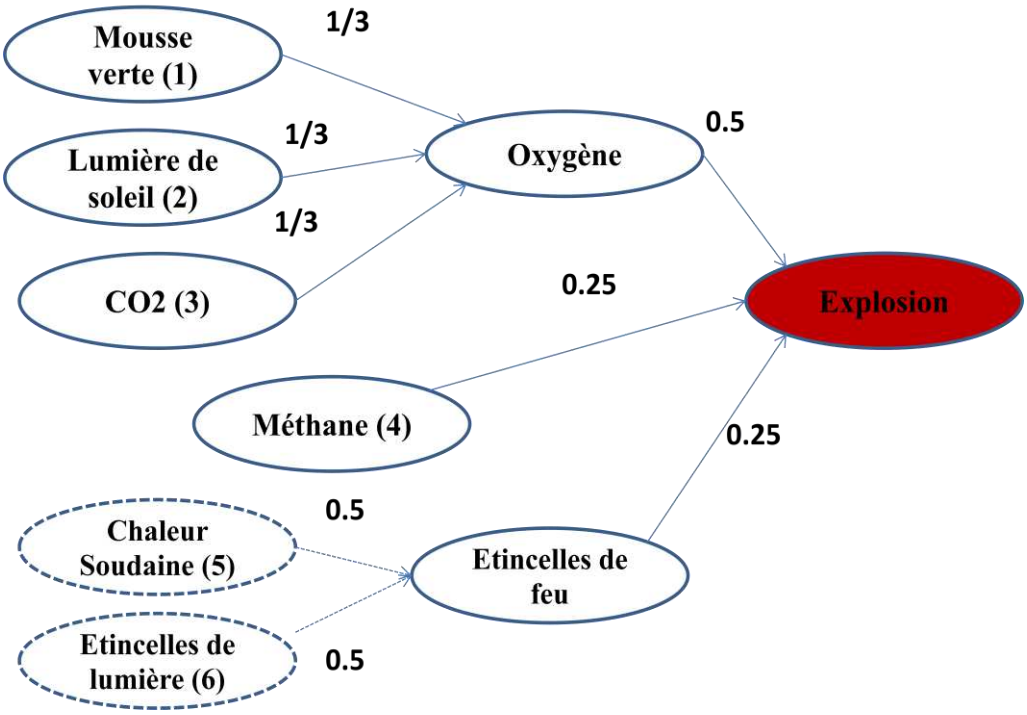


Figure 5-1: Application de RCSF pour la prédiction d'explosion

5.2.1. Services d'application

L'application de prédiction d'explosion, décrite ci-dessus, est une application distribuée de sécurité critique à base d'événements [162], elle offre deux types de services /fonctionnalités/événements :

1) Le service (l'événement) élémentaire est réalisé par un seul capteur et ne peut être décomposé en d'autres événements : la détection de la présence de mousse verte (E1), la détection des rayons solaires (E2), la détection du gaz CO_2 (E3), la détection de méthane (E4), la détection de chaleur soudaine (E5) et les étincelles de lumière (E6).
2) Le service (l'événement) composé est constitué d'au moins deux événements élémentaires ou composés : la détection de taux élevé de méthane (E7), la détection de l'enclenchement du processus de photosynthèse (E8), la détection des étincelles de feu (E9) et le service final de l'application qui est la prédiction de l'explosion (E10).

$$E8 = E1 + E2 + E3 \quad (5.1)$$

$$E9 = E5 + E6 \quad (5.2)$$

$$E10 = E7 + E8 + \text{E9} \quad (5.3)$$

5.2.2. Propriétés de l'application

Comme la plupart des applications de RCSF, l'application de prédiction d'explosion a des propriétés temporelles, spatiales et des dépendances de données qui caractérisent ses services [163].

1. Les *propriétés temporelles* incluent principalement des contraintes temporelles, des propriétés de persistance et de délai associées à certains événements. Exemples :
   - La photosynthèse ne peut être produite que pendant la journée (contraintes temporelles) ;
   - L'information dans le RCSF est valide pour une certaine durée. La durée d'un événement peut être courte, de quelques secondes pour des événements comme les étincelles de feu et chaleur soudaine, ou longue, de deux semaines ou plus, pour des événements comme la présence de mouse (persistance) ;
   - En cas de croissance de mousse verte avec pénétration des rayons solaires en présence de CO2 (E1, E2 et E3), il faut dix minutes pour que l'oxygène soit généré (E4) (délai).

2. Les *propriétés spatiales* associées aux événements sont liées à l'espace géographique qu'ils affectent. Puisque les événements de génération des gaz

(E3 et E7) sont omniprésents, ils occupent un espace plus large que l'événement des étincelles de feu (E8).

3. Les *dépendances des événements* sont réalisées par des fonctions de pondération associées aux événements composés. La fonction de pondération associe une probabilité à chaque événement élémentaire constituant l'événement composé. Par exemple, la photosynthèse (E 8) se produise avec une probabilité de 1/3 de chaque événement élémentaire (E1, E2 et E3).

5.3.Nécessité d'une approche globale de tolérance aux fautes

Le déploiement de prédiction d'explosion sauve plusieurs fois la vie de mineurs et évite de nombreux dommages potentiels dans la mine et les villages voisins. Cependant, les scénarii complexes de panne génèrent des faux positifs et des faux négatifs. Il en résulte que de réelles explosions n'ont pas été détectées et des fausses alarmes générées poussant l'équipe de secours à intervenir inutilement à la mine.

La solution traditionnelle de ces problèmes est de mettre en œuvre un outil de diagnostic efficace pour garder la performance du réseau. Pour cela, une inspection des problèmes du déploiement a montré que les défaillances les plus fréquentes ont été induites principalement par l'épuisement de l'énergie des nœuds et à cause de problèmes de liens de communication. La mise en œuvre d'un outil de diagnostic a contribué à atténuer le problème, mais elle ne peut pas garantir un bon fonctionnement du système à l'exécution. Considérons les cas suivants :

- Le réseau détecte la présence du gaz d'oxygène, sans avoir détecté de mousse verte. Selon la sémantique de l'application, la génération d'oxygène ne peut se produire qu'en cas de présence de mousse verte.
- Le réseau signale le processus de la photosynthèse pendant la nuit. Selon la sémantique de l'application, le processus de la photosynthèse (qui déclenche l'événement de génération d'oxygène) ne peut être produit que pendant la journée.
- Le système génère des fausses alarmes, même dans le cas où l'outil de diagnostic montre des évaluations positives pour l'état des nœuds et des liens.

D'où la nécessité d'identifier efficacement les problèmes potentiels qui affectent les services de l'application, ou le fonctionnement du réseau dans son ensemble. Pour cette raison, nous avons introduit un outil de test d'applications pour compléter l'outil de diagnostic et l'avons intégré à la solution globale de tolérance aux fautes. En effet, l'exigence d'introduire un outil complémentaire et orthogonal à l'outil de diagnostic traditionnel est commune à de nombreuses applications de collaboration complexes [164][165][166] devant traiter des scénarios de panne complexes. Grâce à ces deux outils, les opérations de maintenance seront plus efficaces, conduisant à un système plus fiable et plus performant. C'est l'idée de base de notre solution qui s'appelle « IFTF »(*I*ntegrated *F*ault *T*olerance *F*ramework).

Dans la suite de ce chapitre, nous présenterons l'architecture logicielle de l'approche IFTF, en mettant l'accent sur l'importance du test au niveau de l'application, comme une solution complémentaire à l'outil de diagnostic. Nous illustrerons l'approche par des scenarii de pannes de l'application de prédiction d'explosion décrite ci-dessus sur un modèle de réseau hiérarchique (Figure 5-2).

5.4.IFTF: Architecture Logicielle

Le Framework IFTF est formé de trois composants au niveau du nœud capteur (Figure 5-4), et de quatre composants au niveau du chef de groupe.

1) Le système de validation des données (chef de groupe) ;
2) Le gestionnaire IFTF ;
3) Le service de diagnostic ;
4) Le service de test.

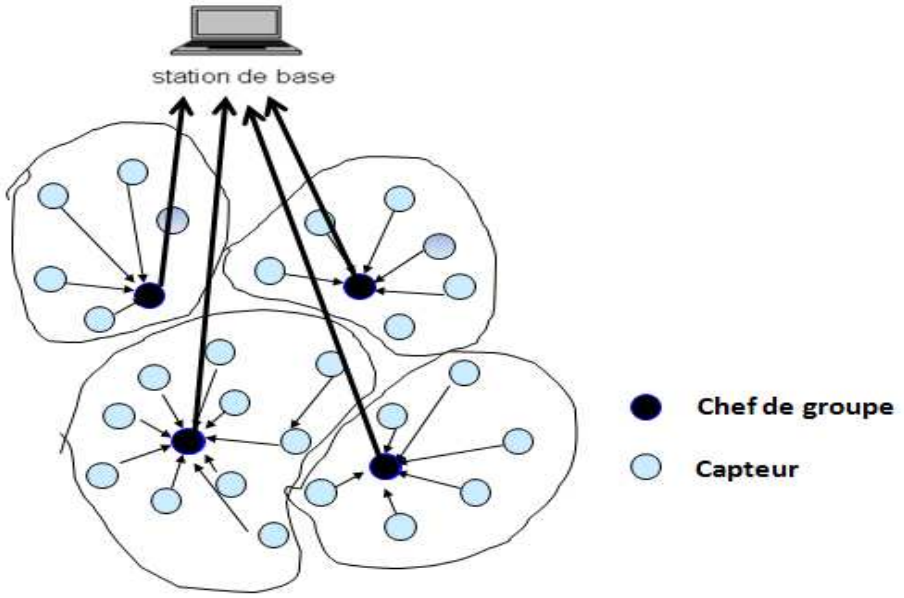


Figure 5-2 : Modèle du réseau

L'ensemble de ces composants (Figure 5-3) permettent de diagnostiquer les pannes du réseau les plus fréquentes, les fautes des données, d'évaluer en temps réel l'impact des fautes sur la performance du système, d'améliorer le taux de détection des fautes cachées (prédéfinies ou silencieuses) et de valider le comportement de l'application suite à des changements dans les conditions d'opérations (reprogrammation, reconfiguration, *etc*.)

5.4.1. La validation des données

Dans le chapitre 3, nous avons proposé une approche de validation des données appropriée à l'architecture hiérarchique de RCSF. Le « cluster head » valide les données en appliquant les méthodes à base de règles et la méthode SOM. Cette méthode permet de

distinguer quatre types des fautes qui sont répandues dans les déploiements de RCSF et permet de réduire le taux des faux négatifs.

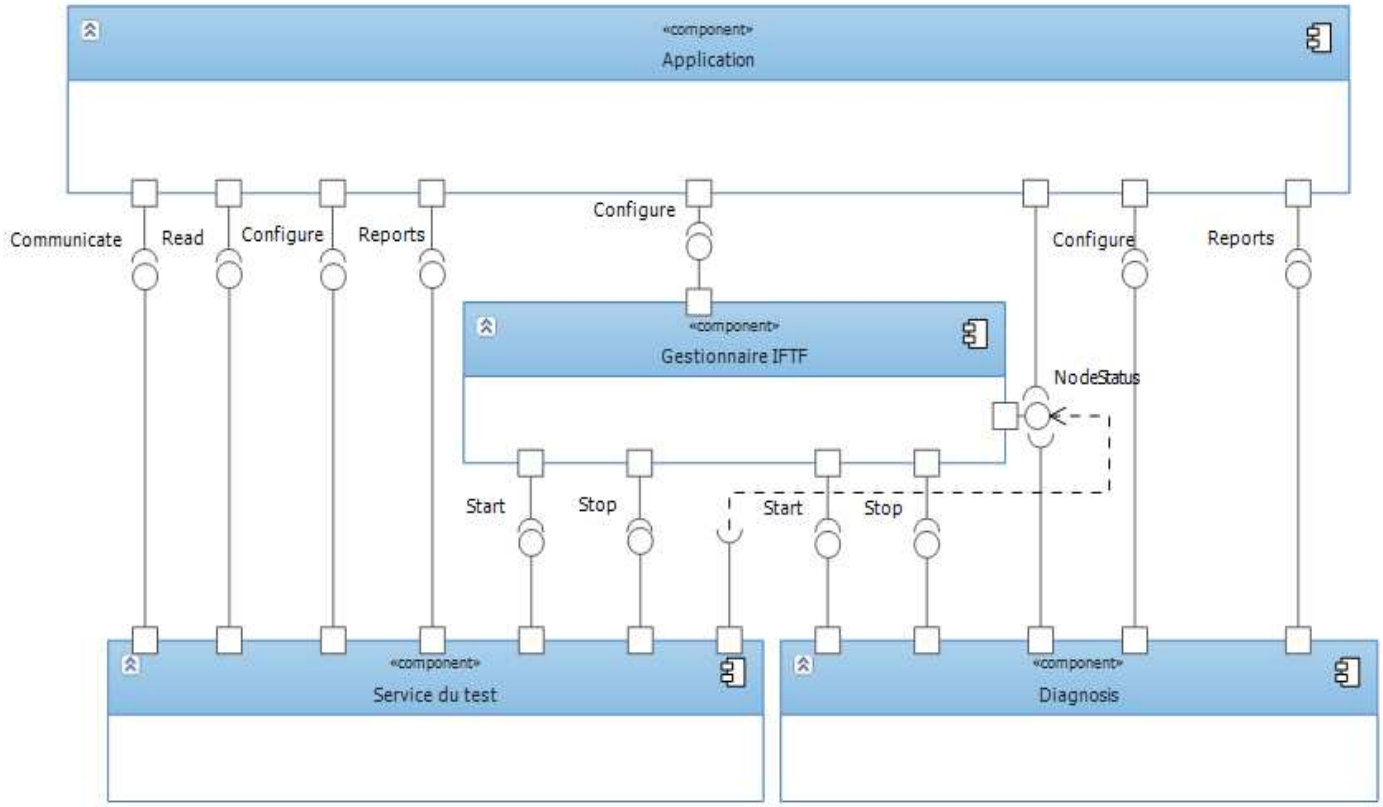


Figure 5-3 : Diagramme des composants de framework IFTF

5.4.2. Le gestionnaire IFTF

L'intégration des deux services test et diagnostic dans un même Framework nous mène à introduire un gestionnaire de Framework qui assure l'interaction des deux

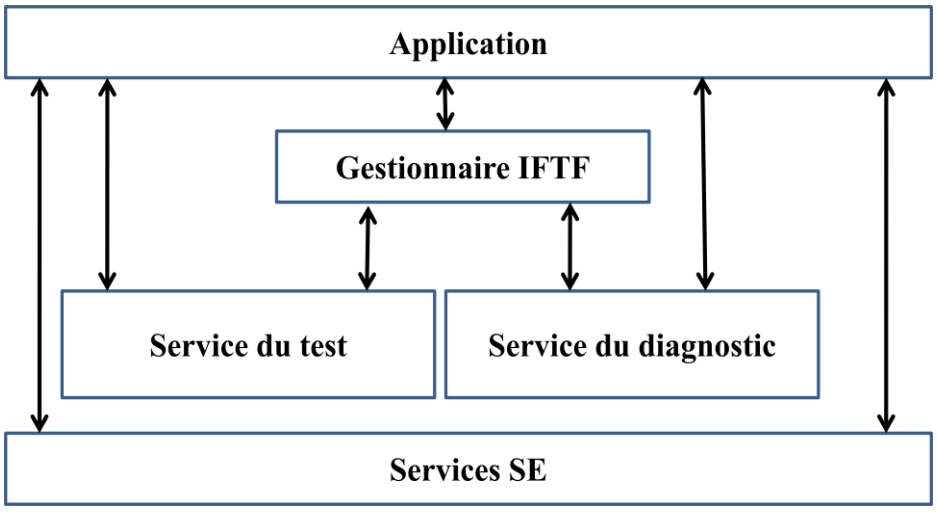


Figure 5-4: Architecture logicielle d'IFTF

services d'une manière efficace (Figure 5-4). Et c'est par une interprétation des résultats de deux services que l'administrateur système peut maintenir un plan de maintenance et de prises de décisions efficaces en interprétant les résultats des deux services.

5.4.2.1. Interfaces du gestionnaire IFTF

Le gestionnaire interagit avec les deux services test et diagnostic, ainsi que la couche d'application à travers deux interfaces fournies et six interfaces requises (Figure 5-3). Les interfaces fournies sont l'interface « Configure » qui permet à l'application de déterminer les périodes de déclenchement du test et du diagnostic et l'interface « NodeStatus » qui informe les composants du système de l'état du nœud (fonctionnement normal, test, diagnostic, test et diagnostic.) (Figure 5-5)

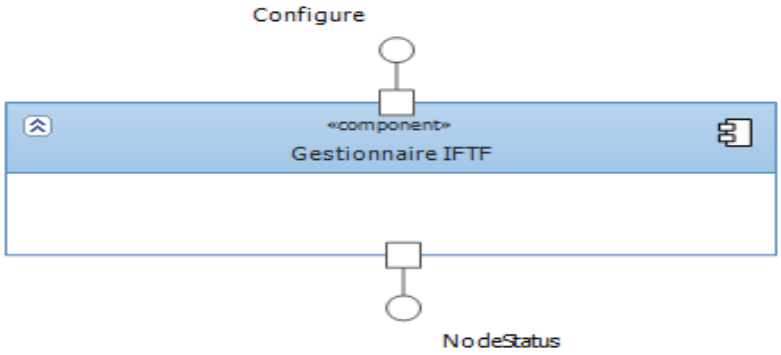


Figure 5-5: Interfaces fournies par le gestionnaire des fautes

Les interfaces requises sont les interfaces « Start », « Stop » de deux services test et diagnostic (Figure 5-6). Le gestionnaire utilise les interfaces « Start » et « Stop » pour lancer et arrêter les services en se basant sur des temporisateurs configurés par la couche application, à la demande de l'administrateur système ou dynamiquement en se basant sur un ensemble de règles définies par la couche d'application ; par exemple, si le nombre de nœuds voisins qui tombent en panne est supérieur à un certain seuil. Dans notre approche, nous considérons tout d'abord le cas des temporisateurs configurés par la couche d'application.

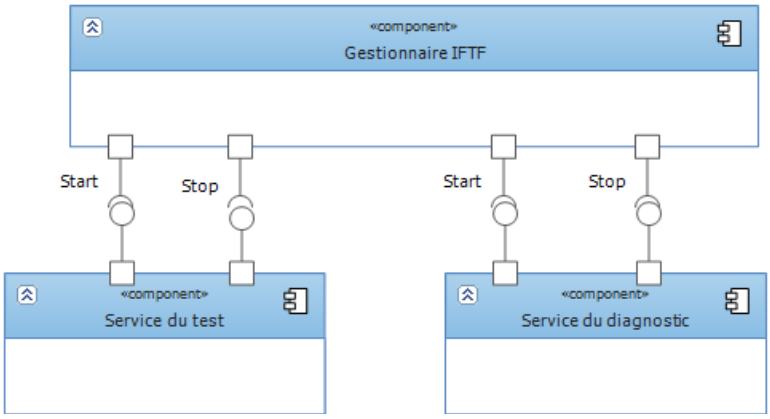


Figure 5-6: Interfaces requises du gestionnaire IFTF

5.4.3. Le service de diagnostic

Le service de diagnostic adopté pour la solution IFTF est le service « SMART » présenté dans le chapitre 4. Du point de vue de la conception, les interfaces fournies et requises par le service « SMART » facilitent son intégration dans le Framework IFTF. Comme nous l'avons décrit dans le chapitre 4, le service « SMART » est un service flexible et léger s'adapte efficacement aux exigences de nombreuses applications RCSF. Le service se focalise sur deux types de défaillances fréquentes dans le monde RCSF : la défaillance des nœuds due à l'épuisement de l'énergie et la défaillance des liens de communication. Le service ne permet pas de détecter tous les types de dysfonctionnements du système. Pour cela, nous avons complété les fonctionnalités offertes par l'outil de diagnostic par un service de test SMART, qui détecte les défaillances au niveau des fonctionnalités offertes par le système à base de RCSF. Avec ces services, l'administrateur aura une vision complète et suffisante pour établir un plan de maintenance efficace.

5.4.4. Le service de test

Notre approche s'inspire d'une solution de test en ligne dans les RCSF [106]. Il s'agit de la solution RTA qui est une approche de test en ligne intégrée à la couche application. L'objectif de notre service de test est similaire à celle de RTA. Cependant, notre approche est conçue comme un service indépendant sous la couche d'application, qui permet de s'adapter aux différentes exigences de l'application (Figure 5-7).

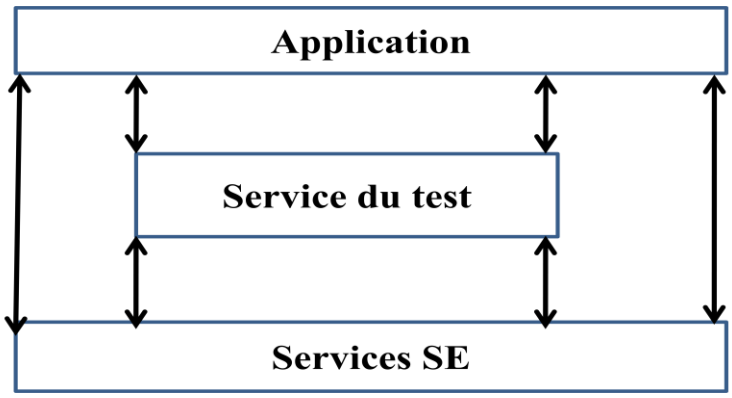


Figure 5-7: Architecture logicielle due service du test

5.4.4.1. Principe de base

Similairement à l'approche RTA [106], notre approche de test se base sur le test fonctionnel. Ce qui motive à adopter ce type de test est qu'il permet de considérer le capteur, comme un composant de type boîte noire. La vérification du comportement correct de l'application ignore la structure interne de ses

composants. Le test fonctionnel permet d'analyser les différentes fonctionnalités ou services offerts par l'application, et de vérifier si ces dernières fonctionnent comme prévu. Les services sont testés en donnant aux nœuds les entrées du test, et en vérifiant si les sorties obtenues sont conformes à celles prévues. Pour cela, l'application contient l'ensemble des cas de tests (associations des entrées et des sorties). Elle émule les réponses des nœuds, ou les notifications envoyées aux nœuds voisins voire au nœud central (par exemple condition d'erreur ou succès).

5.4.4.2. Mise en œuvre

La mise en œuvre du test consiste à injecter dans le réseau des valeurs virtuelles pour simuler le(s) événement(s) réel(s) puis à examiner la sortie et interpréter les résultats. Les valeurs injectées peuvent être positives ou négatives. Par exemple, pour tester le service de détection du taux élevé de méthane, l'injection peut inclure une valeur élevée ou basse représentant la présence ou l'absence du gaz méthane pour vérifier si le service de test fonctionne correctement dans le(s) cas mentionné(s). Pendant le test, les capteurs concernés effectuent tout le traitement, et/ou les communications nécessaires d'une manière transparente dans le réseau. Les réponses du test (positives ou négatives) peuvent être utilisées, afin de mettre en œuvre des stratégies de maintenance efficaces. Plusieurs points sont à considérer quant à la mise en œuvre du test en ligne :

- *Que peut-on tester ?*

  Le test peut inclure n'importe quelle opération qui peut produire des réponses (valeurs ou actions) et peuvent être comparées à celles prévues : les différents types de calcul, les prises de décisions complexes et collaboratives, l'interaction entre les nœuds, *etc*. Le test peut être réalisé de bout en bout (nœuds-puits/station de base) ou de nœud en nœud afin de vérifier une partie spécifique de l'application. Dans notre travail, nous nous focalisons sur le test des événements composés ou les services d'application.

- *Quels nœuds seront impliqués par le test ?*

  La décision dépend de la topologie du réseau et le type de déploiement. Une caractéristique unique d'applications de RCSF est qu'ils sont fortement dépendants de la topologie du réseau. Par exemple, pour tester l'événement de détection de feu dans un bâtiment, nous pouvons tester chaque chambre séparément. Dans les déploiements extérieurs, nous pouvons tester chaque groupe de nœuds « cluster » ou chaque zone géographique indépendamment des autres. De même, dans le cas des réseaux capteurs-actionneurs, un ensemble de nœuds peut être sélectionné pour déclencher une réponse des actionneurs.

- *Quels sont les cas de tests ?*

Réaliser un degré de couverture, sans surcharger les ressources limitées de nœuds de RCSF, peut être effectué par la sélection des cas qui simulent les événements positifs et négatifs. Cela permet de valider l'application avec la capacité courante du réseau. Cela s'avère important surtout pour les événements rares et dangereux (incendie, explosion, *etc*.). Sachant que l'approche RTA a présenté des techniques de réduction des cas de test en se basant sur : le modèle statique de l'application, la nature topologique et la redondance des nœuds.

– *Quand faut-il déclencher le test ?*

Le déclenchement du test peut être effectué d'une manière périodique, à la demande, ou en fonction des critères définis par l'administrateur système, par exemple, les résultats du service de diagnostic. Dans tous les cas, Il est nécessaire de garantir qu'un test est déclenché au moins une fois dans une période donnée, sinon l'apparition de fautes peut passer inaperçue et nuire au fonctionnement de l'application.

– *Quel est le mode de test en ligne le plus favorable à l'application testée ?*

Dans le cas du test en ligne, les processus d'échantillonnage/d'envoi des mesures des capteurs réels et virtuels s'exécutent en parallèle. Une question importante se pose ici : Comment peut-on gérer les interférences entre les tests et les fonctionnalités normales de l'application ? En effet, la gestion traditionnelle des interférences s'effectue selon plusieurs modes [167][168] :

Blocage : Au cours de l'exécution d'un test, les demandes de test ou de l'application sont retardées jusqu'à la fin du processus de test. Les données persistantes de l'application doivent être restaurées à l'état original après la fin du processus de test.
Abandon du processus de test : Lorsqu'un autre composant demande la fonctionnalité normale du composant testé, le processus de test est abandonné. Une fois abandonné, les données du composant doivent être restaurées.

Il est important de noter qu'aucune solution n'est optimale et que le choix d'un mode dépend du type d'application considérée. Quand les applications considérées sont critiques, les demandes de test sont abandonnées au profit des appels fonctionnels. Pour ce type d'applications, nous introduisons la notion de données urgentes pour décider l'abandon du test. Quand les applications ne sont pas critiques, il est envisageable d'interrompre momentanément les fonctionnalités de l'application.

– *Le coût du test* Etant donné les ressources limitées des nœuds capteurs, notre approche permet de sélectionner les services potentiels à tester (énergie), les cas de tests (mémoire), le temps de déclenchement des tests (énergie) ainsi que les nœuds concernés par le test (énergie). Notre approche permet aussi de tester un à un les différents services attendus par l'utilisateur final, et permet ainsi d'adopter les stratégies de maintenance les plus favorables en fonction des résultats du test (Figure 5-8). De plus, pour minimiser la consommation d'énergie, nous introduisons une politique de dépendance entre les tests. La dépendance entre les tests s'applique principalement sur les services composés. Lorsque nous déclenchons un test concernant un service composé, et si le résultat du test est négatif, nous pourrons déclencher les tests pour les services élémentaires qui le constituent. Par exemple, si le résultat du test de service de prédiction d'explosion est négatif, on peut tester les services contribuant à son déclenchement un à un successivement. Ainsi, on peut tester le service de détection du méthane et ainsi de suite. Cette stratégie de tests successifs permet de réduire l'énergie consommée pendant le processus de test. Cependant, la stratégie nécessite un ensemble de règles de dépendance entre les types de tests pour les déclencher efficacement.

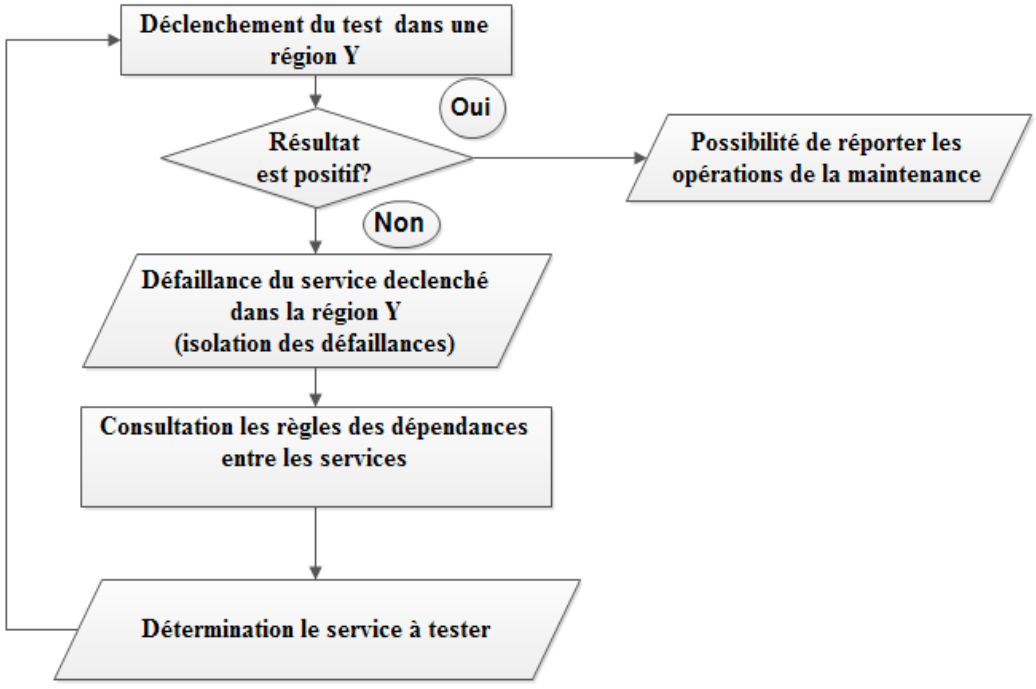


Figure 5-8: Interprétation les résultats du service du test

5.4.4.3. Interfaces du service de test

Le service de test interagit avec la couche utilisatrice (l'application ou le gestionnaire IFTF) à travers des interfaces bien définies. Il fournit six interfaces

aux couches supérieures (Figure 5-9) et requiert deux interfaces de composants du système d'exploitation (Figure 5-10).

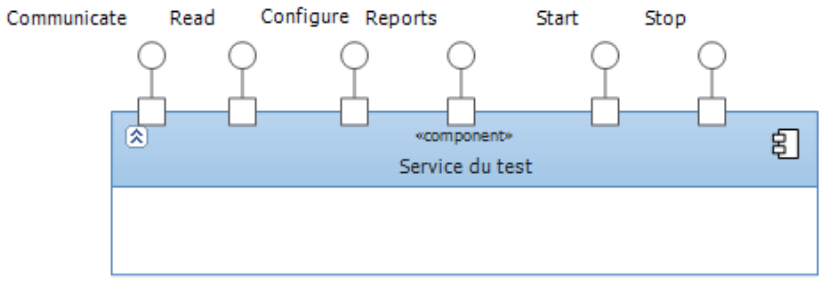


Figure 5-9: Interfaces fournies du service de test

E. Interfaces fournies

Les interfaces fournies sont les interfaces de configuration, de déclenchement, de communication et de notification.

1) L'interface « Configure» : fournit les éléments suivants à la couche utilisatrice (application) pour déterminer et ajuster en ligne les paramètres du service :
    a. Les cas de tests : ils représentent l'ensemble des entrées/sorties qui simulent les événements virtuels au sein du réseau. Ils dépendent de l'application.
    b. Le mode de test : blocage, abandon du test en cas d'urgence ou commutation.
    c. La valeur urgente : la valeur urgente entraîne l'abandon du test pour continuer le fonctionnement normal de l'application.
2) L'interface « Start » : permet à la couche utilisatrice (le gestionnaire IFTF) de déclencher un test spécifique. Quand une session de test commence, cette commande modifie le statut du nœud en mode test. Par défaut, la commande déclenche un test de type bout à en bout (du nœud au puits).
3) L'interface « Stop » : permet d'arrêter un test spécifique. Elle modifie le statut du nœud en statut normal d'opération.
4) L'interface « Read » : permet à la couche application de lire une nouvelle valeur. Selon le statut courant du nœud (mode de test ou mode normal), la commande retourne une valeur virtuelle ou réelle. Dans le cas du mode normal d'opération, l'application reçoit un champ de validité indiquant la validité de la valeur (valeur aberrante). L'application peut filtrer les messages en fonction de sa propre politique (avertissement immédiat, ou notification retardée). Ce champ permet donc une optimisation de la consommation énergétique.

5) L'interface « Receive » : permet à l'application de recevoir les paquets du test et les paquets réels d'une manière transparente.
6) L'interface « Report » : fournit à la couche application le compte rendu d'erreurs.

F. Interfaces requises

Le service de test requiert deux interfaces du système d'exploitation (Figure 5-10) et une interface du gestionnaire IFTF :

1) L'interface du composant de pilote du capteur pour lire les valeurs réelles.
2) L'interface du composant communication pour recevoir et envoyer les messages du test.
3) L'interface « NodeStatus » du gestionnaire IFTF pour renseigner sur l'état du nœud.

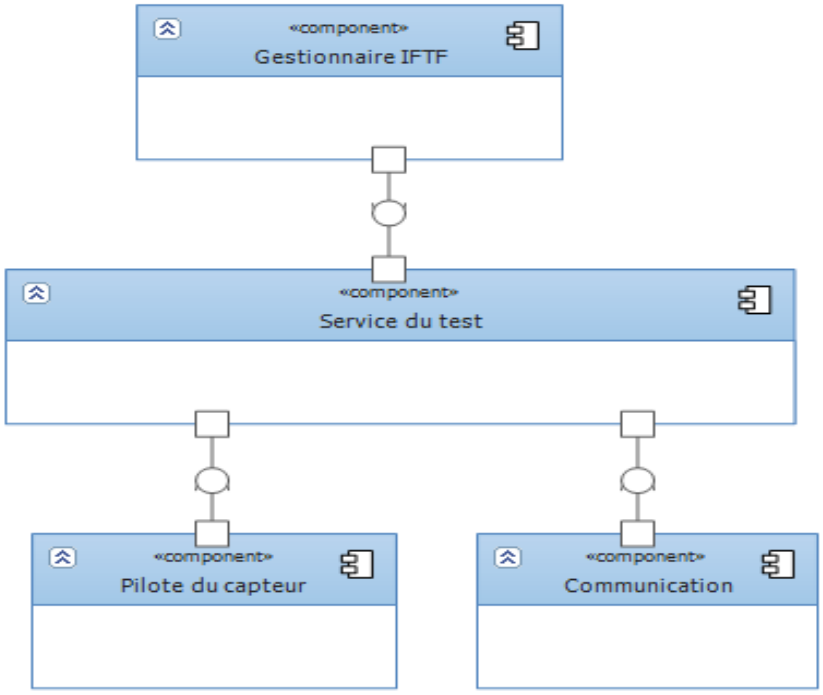


Figure 5-10: Interfaces requises par le service du test

5.4.4.4. Les composants du service de test

Lorsqu'une session de test commence, l'application lit les valeurs virtuelles et marque (insère un champ indiquant que le message est de type test) les messages d'application envoyés par un nœud. Lorsque la session se termine, l'application retourne à l'état de fonctionnement normal en restaurant tous les paramètres du

capteur à leur état initial. Pour réaliser les fonctionnalités de test, notre service inclut les quatre composants suivants : le composant de capteurs virtuels, le gestionnaire d'entrée, le gestionnaire de la communication et le notificateur des résultats (Figure 5-11).

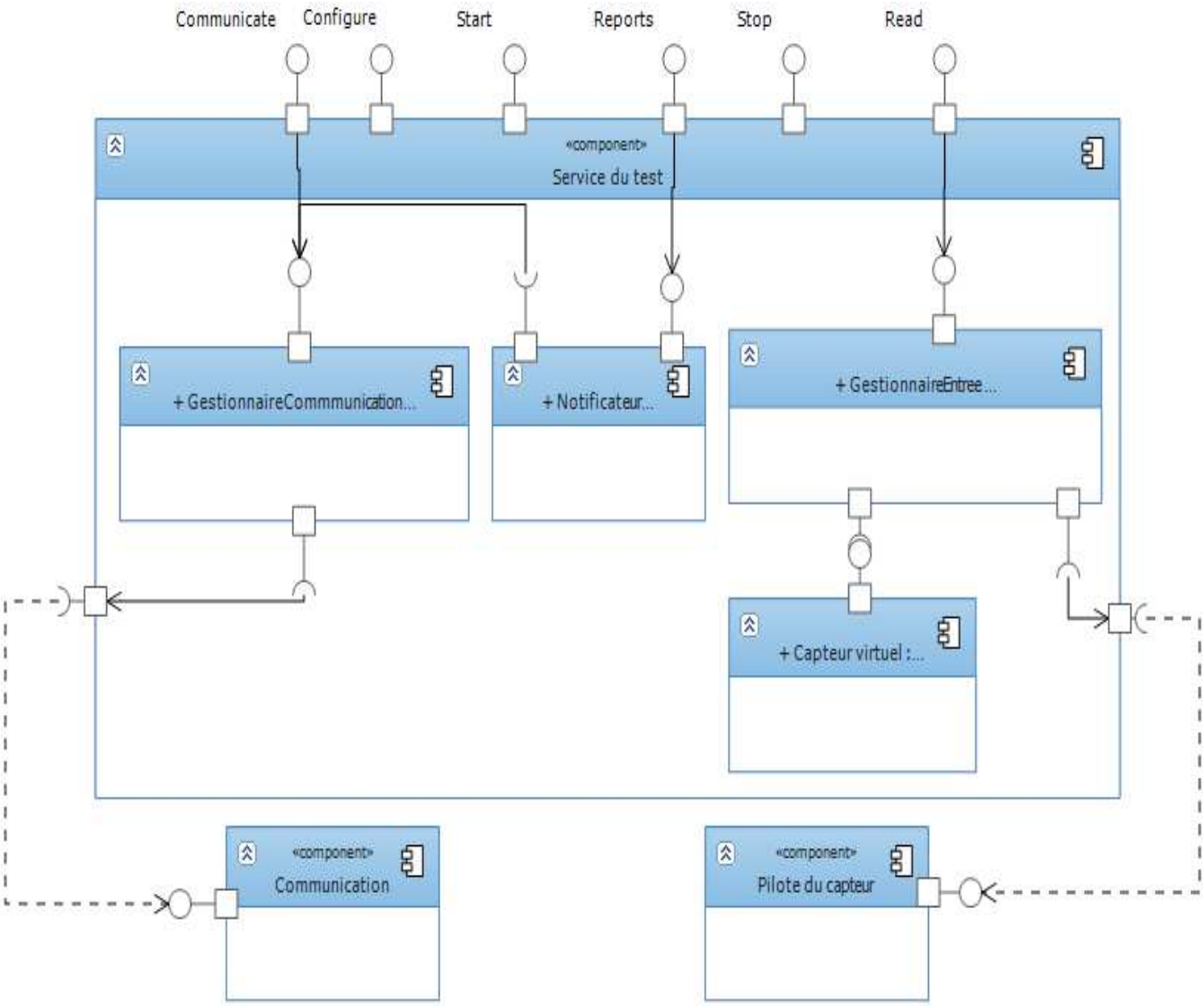


Figure 5-11: Diagramme des composants du service de test

– **Le composant des capteurs virtuels** : ce composant gère des tampons mémorisant les cas de tests et pouvant être échantillonnés de la même manière que les capteurs réels. Ce composant peut gérer plusieurs types de capteurs virtuels, par exemple, capteur de température, capteur de détection de présence de mousse verte, *etc*.

- **Le gestionnaire d'entrée** « InputHandler » : ce composant gère les entrées fournies à la couche application. Selon le statut courant du nœud (test ou fonctionnement normal), le gestionnaire interagit avec les capteurs physiques (mode normal) ou les capteurs virtuels (mode de test). Selon le statut courant du nœud et le mode de test, il renvoie à l'application les données réelles ou virtuelles. Lorsque l'application lance une demande de lecture d'un capteur du nœud, nous obtenons alors plusieurs possibilités :

  1) Si le nœud est en état de fonctionnement normal, le gestionnaire des entrées redirige la demande de lecture vers les pilotes des capteurs.
  2) Si le nœud est en état de test, le gestionnaire examine le mode de test, s'il est en mode blocage, il redirige la demande de lecture vers les capteurs virtuels. Sinon, il redirige la demande vers les capteurs virtuels et réels, et en cas de données urgentes des capteurs réels, il abandonne le test et passe au mode de fonctionnement normal du système.

- **Le gestionnaire de communication** « CommunicationHandler » : il marque les paquets pour distinguer les paquets de test des paquets réels de l'application. Il permet aussi de vérifier l'état des voisins (session de test ou non) en examinant les marques à la réception des messages de l'application.

- **Le notificateur :** Il compare les sorties de test aux sorties prévues afin d'examiner les résultats du test.

5.4.4.5. Format du message de test

Le message de test contient les cinq champs suivants (Figure 5-12) :

- Type (1 octet) : ce champ indique le type du message (réel ou virtuel)
- Estampille de temps (4 octets) : indique la date d'envoi le message.
- Numéro de séquence (4 octets) : indique le numéro du message. En examinant la continuité du numéro de séquence, le nœud peut détecter la perte du paquet et demande sa retransmission.
- Emetteur (2 octets) : indique l'identifiant de l'émetteur du message. Normalement, les champs de l'émetteur et le numéro de séquence sont ajoutés par l'application par défaut.
- Mesure (4 octets) : ce champ représente la mesure virtuelle du test ou réelle de l'application.

Type (1 octet)	Emetteur (2 octets)	Numéro de séquence (4 octets)	Estampille (4 octets)	Mesure (4 octets)

5.

Figure 5-12 : Format du message de test

Afin d'illustrer l'avantage d'intégrer les deux services mettant en œuvre des plans de maintenance efficace, considérons l'application de prédiction d'explosion où

l'application fournit plusieurs types de services, par exemple, la prédiction d'explosion, la détection d'oxygène, la détection de taux élevé de méthane, *etc*. La solution IFTF peut répondre aux questions comme, par exemple :

- *Est ce que le service de prédiction d'explosion fonctionne correctement dans la mine ? Sinon, quelles sont les causes de dysfonctionnement ?*
- *Est-ce que le service de détection d'oxygène fonctionne correctement dans la région X ?*
- *Est-ce que le système peut répondre aux exigences de qualité des services ou les métriques de performances globales (latence de notification, fiabilité, consommation d'énergie, taux de faux positifs ou négatifs, etc.) attendues par l'utilisateur final ?*
- *Est–ce que chaque service fonctionne correctement malgré les fautes, les changements d'environnements, la reconfiguration, etc.? Est-ce que le système est bien configuré et adapté à l'environnement où il a été déployé ? etc.*

Scenario 1 : Service de détection de la présence d'oxygène pendant la journée
Description : Un « cluster Head » notifie le nœud central qu'il détecte l'oxygène (l'événement E7) dans le cas où il reçoit des valeurs positives du capteur de mousse verte (l'événement E1), du capteur de gaz CO_2 (l'événement E2) et celui de détection des rayons solaires (l'événement E3). E1, E2 et E3 doivent êtres produits pendant la journée car le processus de la photosynthèse ne peut se produire que pendant la journée. Prenons maintenant, les cas suivants :

- Cas 1 : La panne d'un nœud /capteur
  Si S1 tombe en panne (Figure 5-13), le service de diagnostic traditionnel peut détecter ce type de fautes et notifier ainsi le nœud central pendant la période de notification des fautes. Cependant, par rapport au service de test, la panne de S1 ne sera pas considérée comme un problème car il y a un capteur redondant au sein du « cluster ». Par contre, la panne du capteur S2 est considérée comme un problème pour le service de test puisqu'elle affecte le service de détection d'oxygène. De cette façon, l'approche de test permet de réduire le taux de faux positifs, en ignorant la panne des éléments, car il y a une redondance qui tolère la panne du capteur, c-à-d la panne du capteur dans ce cas ne nuit pas aux services de l'application.
- Cas 2 : La mobilité d'un nœud/capteur
  Supposons que la localisation de S3 se transpose à un autre « cluster ». Ici, la mobilité de S3 représente un problème au niveau de la sémantique de l'application. Dans ce cas, le service de test envoie une alerte au nœud central puisqu'il détecte un dysfonctionnement du service de détection d'oxygène. Les approches traditionnelles de diagnostic peuvent ignorer la mobilité de S3 comme si le nœud ou le capteur était encore en bon état.

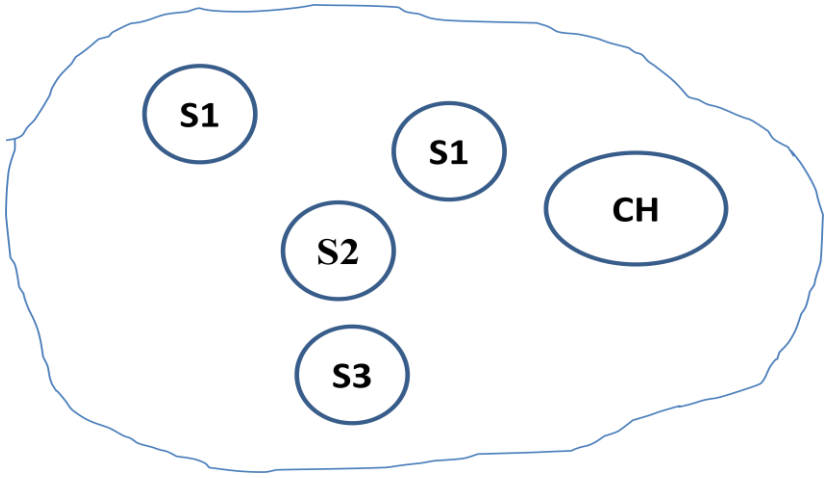


Figure 5-13: Topologie d'une zone X

- Cas 3 : La déviation d'horloge
  La déviation d'horloge représente aussi une faute au niveau de la sémantique de l'application. Supposons que l'horloge du nœud « cluster head » a subi une déviation significative retard/avance de 12 heures en moyenne. Dans ce cas, même si le « cluster head » reçoit des valeurs positives concernant les événements E1, E2 et E3 pendant la journée, il ne notifiera pas au nœud central l'occurrence de l'événement E8, car selon la sémantique de l'application, l'événement E8 (processus de photosynthèse) doit se produire pendant la journée et non pas pendant la nuit comme l'horloge du « cluster head» l'indique. L'événement virtuel créé lors du lancement du test permet de détecter un dysfonctionnement dans le service de détection d'oxygène. Les cas 2 et 3 montrent que l'approche de test peut réduire le taux des faux négatifs en dévoilant des problèmes cachés aux services de diagnostic traditionnels.

Scenario 2 : Service de détection des étincelles de feu

Description : Le « cluster head » notifie le nœud central qu'il détecte l'événement des étincelles de feu (E9) si les deux événements E5 et E6 se produisent. Supposons que ce service exige que l'événement E7 doit être notifié par au moins deux capteurs de type S7 (Figure 5-14). Supposons que selon la sémantique de l'application, la persistance de l'événement E7 est de 3 secondes.

- Cas 1 : La panne du capteur
  La panne du capteur entraîne l'arrêt du service au sein du « cluster ».
- Cas 2 : Le retard de notification

Considérons le cas ou le « cluster head » reçoit une valeur positive concernant l'événement E6, puis après 4 secondes l'événement E8 (Figure 5-14). Le retard de notification dans ce cas affecte le service de détection des étincelles de feu. Le service de test peut détecter le dysfonctionnement du service et l'administrateur système peut prendre alors des décisions concernant la reconfiguration, ou la reprogrammation de l'application.

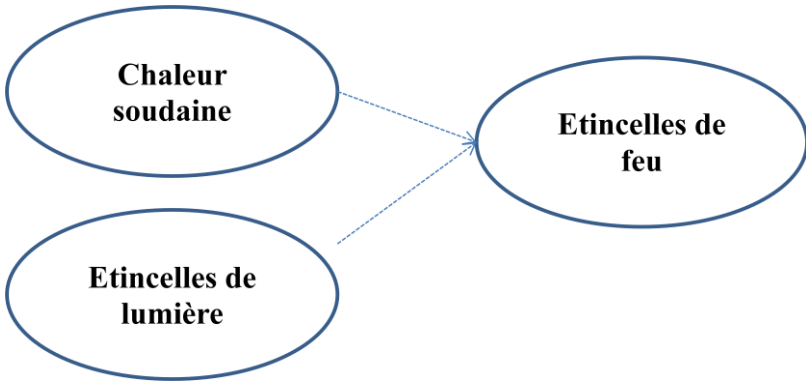


Figure 5-14 : Service de détection des étincelles de feu

Scenario 3 : Service de prédiction d'explosion

Description : Un « cluster head » avertit l'utilisateur final qu'il y a un risque d'explosion (événement E10) dans une certaine durée, dès qu'il reçoit une affirmation émanant de trois services : la détection d'oxygène (l'événement E8), le taux élevé de concentration du méthane (l'événement E7) et la détection des étincelles de chaleur (l'événement E9) (Figure 5-14).
Ce scénario présente des opérations complexes (interactions entre les nœuds, prise de décision collaborative, échange de messages, *etc*.) pour réaliser le service de prédiction. Cette complexité rend la solution IFTF plus bénéfique En effet, l'interprétation des résultats du test et du diagnostic profite du processus de *l'isolation des fautes* au niveau géographique et au niveau du service, ainsi que des décisions pertinentes au niveau de la maintenance (Figure 5-15).

Lorsqu'on déclenche un test pour un service X dans une région Y, il y a deux possibilités :

- Si le résultat du test est positif, l'administrateur système peut reporter les opérations de maintenance, même si le service de diagnostic signale qu'il y a des fautes. Ceci provient principalement de la nature de la tolérance aux fautes inhérente au système de RCSF (chemins redondants d'un protocole de routage, nœuds redondants, *etc*.) ;
- Si le résultat du test est négatif, l'administrateur doit vérifier l'état des nœuds et des liens par le service de diagnostic dans la région Y (isolation des défaillances au niveau géographique). Dans le cas où le service de diagnostic indique qu'il n'y

a pas de problèmes au niveau des éléments du système, on peut en déduire qu'il y a un dysfonctionnement du service dû à des fautes qui ne sont pas détectables par le service du diagnostic. Après une consultation des règles de dépendances entre les services, l'administrateur décide quel sera le prochain service à tester, et *ainsi de suite.* Des inspections doivent être effectuées pour vérifier la bonne configuration des paramètres du système, en passant par la validation de l'application, la capacité du réseau, *etc*.

Finalement, les deux outils peuvent ensemble fournir un compte-rendu détaillé sur les problèmes potentiels et marginaux du système. Par conséquent, l'administrateur peut mettre en œuvre un système de recouvrement le plus convenable et efficace pour garantir un niveau de qualité de service à l'utilisateur final.

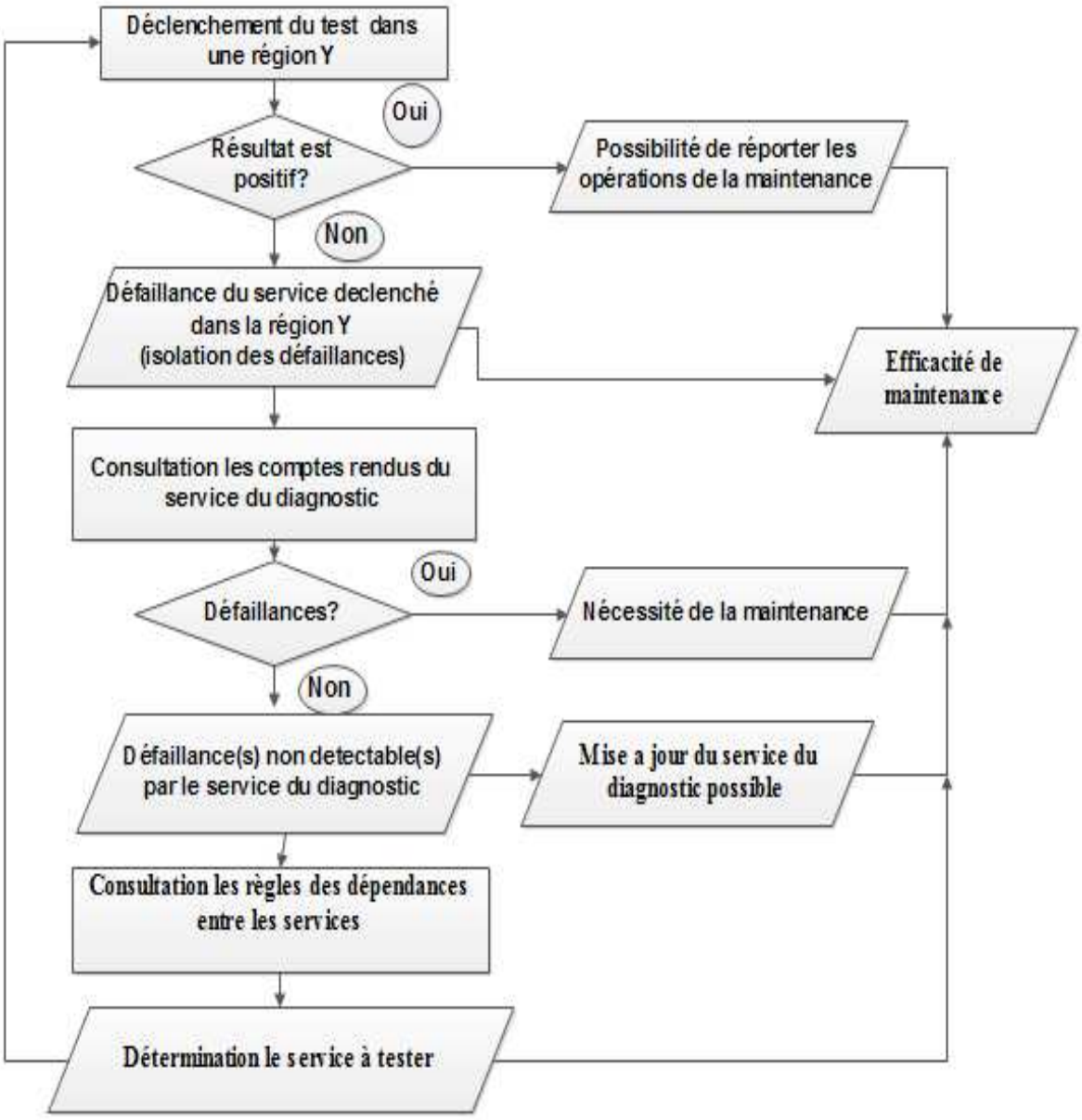


Figure 5-15: IFTF pour une maintenance plus efficace

5.5.Evaluation du Framework IFTF

Nous avons étudié trois critères de performance pour évaluer le Framework IFTF : coût mémoire, coût énergétique et le taux des faux négatifs. Le coût mémoire est une métrique importante dans notre approche, étant donné que les composants d'IFTF résident sur chaque nœud du réseau. Une métrique de performance aussi importante dans le RCSF est la consommation d'énergie. Comme 70% de la dissipation d'énergie est due à la surcharge de la communication [159], nous avons focalisé sur le coût de la communication induite par les services du système. Pour évaluer la performance de la solution, nous avons utilisé le simulateur Tossim[93] qui fonctionne avec le système d'exploitation TinyOS[21]. TOSSIM, compile directement depuis le code TinyOS. TOSSIM peut simuler des milliers de nœuds simultanément. TOSSIM fournit « run-time » la sortie de débogage configurable, permettant d'examiner l'exécution d'une application à partir de différents points de vue sans avoir à recompiler. L'application TinyOS et les modules de TOSSIM sont compilés et liés à travers une librairie. Cette librairie ainsi qu'un interpréteur Python, sont utilisés pour définir la topologie du réseau, pour configurer et lancer la simulation.

5.5.1. Coût mémoire

L'espace mémoire occupée par le service de diagnostic sur la plateforme TinyOS est d'environ 13 Kbytes en ROM et 1 Kbytes en RAM (chapitre 3). L'espace mémoire occupée par les composants de test est d'environ 20 Kbytes en ROM et 2 kbytes en RAM. En effet, l'implémentation du service inclut principalement une table d'association des entrées-sorties virtuelles, un temporisateur pour planifier le temps de déclenchement des tests, les gestionnaires des entrées et des sorties. La taille de la table dépend du nombre de services à tester et des cas de tests pour chaque type de service. Les cas de tests ont été sélectionnés pour ne pas surcharger les ressources des nœuds. Pour cela, nous avons choisi une valeur positive et une valeur négative pour tester une fonctionnalité ou un service de l'application. Avec ces valeurs, il est possible de valider l'application pendant le déploiement pour vérifier qu'elle est toujours adaptée à l'environnement dynamique du RCSF. Concernant les gestionnaires d'entrées et communications, ils font uniquement appel à des interfaces existantes pour générer les événements ou les services simulés. Certains événements peuvent exiger de mémoriser certains paramètres de services. Dans ce cas, le coût mémoire augmente pour fournir une validation plus exacte.

5.5.2. Coût énergétique

Nous avons étudié le nombre de messages afin d'estimer le coût énergétique. Le diagnostic induit un échange des messages pour détecter les éléments défaillants et identifier la cause d'une défaillance. Le test induit un échange pour créer les événements virtuels. Pour étudier l'effet de la redondance, nous avons varié le nombre de nœuds par « cluster ». Le test est déclenché par un temporisateur à base de rotation. L'expérimentation dure dix minutes, durant laquelle un test pour l'événement des étincelles de feu a été lancé dans chaque cluster à part et un test pour le service de

diagnostic à chaque minute. La charge de la communication du service de test s'accroît légèrement lorsque le nombre de nœuds dans chaque « cluster » augmente, ceci étant dû principalement à la communication multi-saut. La Figure 5-16 montre que le Framework IFTF induit un surcharge de 4% en comparaison au fonctionnement du service de diagnostic uniquement.

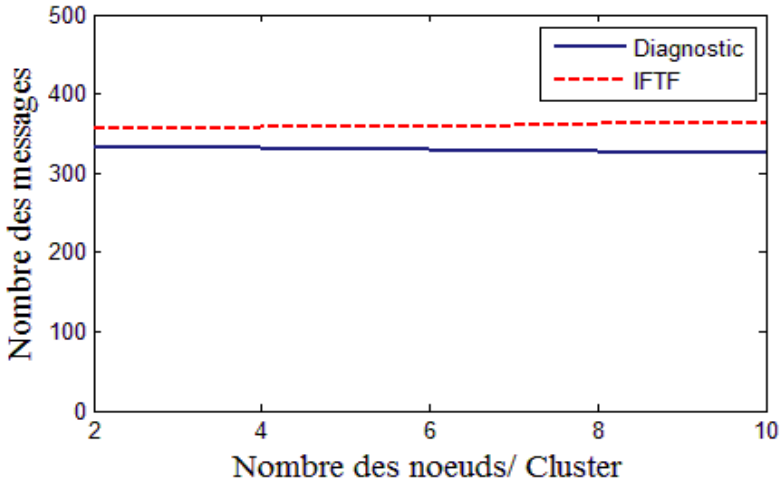


Figure 5-16 : Coût énergétique d'IFTF et le service de diagnostic

5.5.3. Taux des faux négatifs

Nous avons étudié l'exactitude de la détection pour le service de détection d'étincelles de feux en calculant le taux des faux négatifs de deux approches : le diagnostic et L'IFTF. Le taux est calculé par la formule suivante :

$$\text{Taux des faux négatifs} = \frac{\text{Nombre des feux non détectés}}{\text{Nombre des feux générés}} \quad (5.1)$$

Selon la sémantique de l'application, deux nœuds doivent signaler la présence de feux pour que le réseau signale la présence d'étincelles de feux. Nous avons étudié les performances de la solution selon deux types de défaillance :

- Défaillance au niveau des composants du système : ce type représente des défaillances isolées ou indépendantes des nœuds. Pour simuler les défaillances isolées, on a déconnecté des nœuds choisis au hasard avec un taux de 1 nœud/seconde. La déconnexion d'un nœud est simulée par la suppression de tous ses liens dans le modèle de radio TOSSIM. Les nœuds déconnectés, ont été reconnectés après une notification d'une défaillance des nœuds dans le réseau.
- Défaillance au niveau de la sémantique de l'application : pour ce type de défaillance, nous avons étudié l'erreur de mobilité. Dans notre cas, la mobilité

d'un nœud peut entraîner une défaillance au niveau de l'application si le nombre des nœuds présents dans un cluster devient inférieur à deux. Pour simuler les défaillances de l'application, nous avons changé la localisation des nœuds, choisis au hasard avec un taux de 1 nœud/seconde, dans des clusters adjacents.

Les résultats expérimentaux montrent que les taux des faux négatifs du service de diagnostic, du service de test et de l'approche IFTF sont similaires pour la défaillance des nœuds pour les différentes tailles de cluster (Figure 5-17). Tandis que pour l'erreur de mobilité, la Figure 5-18 montre que l'approche IFTF a réduit le taux des faux négatifs de 60% en comparaison du service de diagnostic. En fait, le changement de la localisation d'un nœud n'empêche pas les messages du protocole de diagnostic d'arriver à ses voisins dans le cluster d'origine. Ces derniers le considèrent comme il étant encore opérationnel. Cependant, l'approche IFTF, à travers son sous-service de test, a détecté le problème d'erreur de mobilité puisqu'il est lié a la sémantique de l'application. Lorsque le nombre de nœuds présents dans un cluster est égal à 5, le taux des faux négatifs devient nul. En se basant sur les résultats expérimentaux, L'IFTF permet une visibilité de l'état du système plus pertinente que l'outil de diagnostic en détectant des problèmes potentiels au niveau du comportement de l'application et en induisant un surcoût acceptable des ressources.

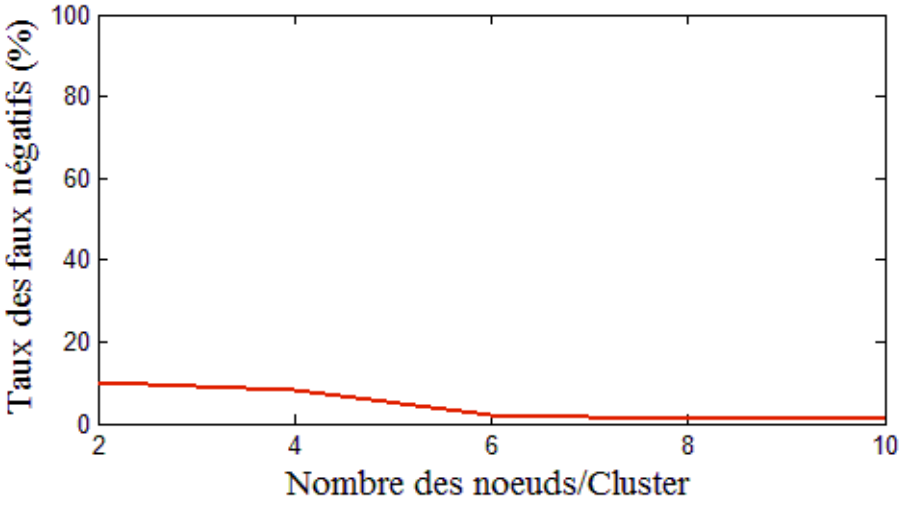


Figure 5-17 : Taux des faux négatifs pour les deux approches diagnostic et IFTF (défaillances des nœuds)

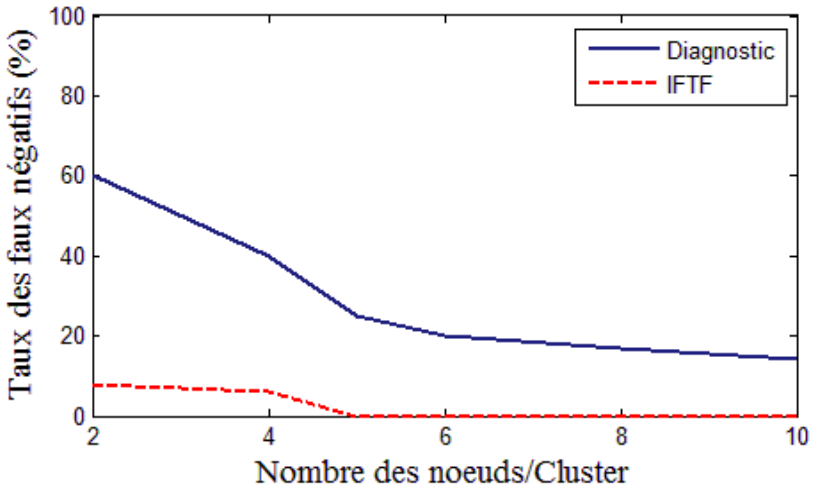


Figure 5-18 : Taux des faux négatifs pour les deux approches diagnostic et IFTF (erreur de mobilité)

5.6. Conclusion

Les approches de tolérance aux pannes traditionnelles mettent l'accent sur certains états d'incohérence du système. Par exemple, le système peut observer une perte de données et la diagnostiquer en identifiant si elle provient de fautes logicielles, de pannes de nœuds, de problèmes de liaisons, *etc*. Le système de détection peut également détecter une faute logicielle, et la diagnostiquer en déterminant si elle provient d'un débordement de la pile, d'une concurrence critique, *etc*. Donc, nous pouvons avoir différents niveaux de diagnostics. Chaque problème peut être détecté et diagnostiqué à ces niveaux différents. Cependant, plus le diagnostic est minutieux, plus la consommation d'énergie sera élevée. En effet, le nombre d'éléments à surveiller avec leur degré de surveillance est étroitement corrélé à la consommation d'énergie. Cependant, notre approche, en combinant le service de test d'applications et le service de diagnostic, permet une surveillance du système à différents niveaux sans augmenter significativement la consommation d'énergie. En outre, elle fournit des informations plus utiles à l'administrateur réseau : quelles fautes affectent le service d'application, quelles fautes doivent être réparées rapidement (car elles affectent les performances du système) et quelles fautes peuvent être ignorées, même temporairement.

Chapitre 6.

CONCLUSIONS ET PERSPECTIVES

Les réseaux de capteurs suscitent un intérêt croissant étant donné la diversité de leurs applications. Actuellement, cette large étendue d'applications fait que les réseaux de capteurs sont une partie intégrante de notre vie. Cependant, la réalisation d'une application à base de RCSF pose de nombreux défis liés aux enjeux de la sûreté de fonctionnement de ces systèmes. Dans ce contexte, d'une manière générale, cette thèse a pour but de contribuer à la conception d'une approche globale et générale de la gestion des fautes à grande échelle des capteurs.

SOMMAIRE

6.1. Rappels des objectifs

Nous rappelons ici brièvement les objectifs de cette thèse qui s'articulent autour des trois points suivants.

- **Architecture globale :** Un comportement correct d'un système à base de RCSF est lié principalement à trois aspects fondamentaux : la fiabilité des mesures collectées de l'environnement, la fiabilité des différents éléments qui entrent dans la composition d'un tel système (liens de communication et nœuds) et la fiabilité de l'application finale. Notre but est de proposer une approche globale traitant ces aspects en tenant compte du facteur de coût (énergie et mémoire).
- **Architecture générale :** L'hétérogénéité au niveau des applications liées aux réseaux de capteurs sans fils, est une réalité qui doit être prise en compte par les solutions visant à être génériques et polyvalentes. Notre but est de proposer une approche le plus possible indépendante des types d'applications et des types de capteurs utilisés.
- **Passage à l'échelle :** Le nombre croissant de capteurs et d'applications utilisant des capteurs rend critique le développement de solutions qui peuvent résister à un nombre croissant des paquets de contrôle. En considérant ce fait, notre solution vise une inspection des fautes à grande échelle.

En considérant ces objectifs, le résumé des contributions de cette thèse est présenté dans la section suivante.

6.2. Contributions

La principale contribution de cette thèse a été de proposer une approche *globale* pour traiter les aspects de couverture de fautes en tenant compte du coût des ressources. Notre approche IFTF adresse les fautes à trois niveaux : données, nœud/réseau et application. Nous avons traité séparément chaque classe de faute en tenant compte des enjeux d'énergie, de l'exactitude de détection, de la flexibilité et de l'hétérogénéité des capteurs et applications.

6.2.1. Détection et classification des fautes de données

Les données de capteurs sont souvent hétérogènes, collectées dans des situations souvent difficiles, possèdent des caractéristiques spatiales et temporelles, et évoluent au cours du temps. De ce fait, nous avons proposé un système hybride fondé sur deux méthodes complémentaires : méthode à base de règles et méthode d'apprentissage non supervisée sur la base de la corrélation spatiale. Notre approche se caractérise par la possibilité de classifier les fautes en quatre types (faute brusque, faute de type bruit, faute bloquante et valeur aberrante), de réduire le taux des faux négatifs. Nous avons étudié l'efficacité de chaque méthode par une étude expérimentale portée sur une base de données réelle. L'approche proposée n'induit pas une surcharge de la communication. L'espace mémoire n'est pas une préoccupation dans le cas de l'architecture hiérarchique du réseau. Cette architecture nous permet d'implémenter le système de détection des fautes au niveau du « clusterhead ». Grâce à ce système de

détection des fautes, IFTF peut filtrer les données erronées nuisant aux prises de décisions ou à la mauvaise interprétation des événements du monde réel.

6.2.2. Diagnostic des fautes au niveau nœud/réseau

Les problèmes des liens et d'épuisement de la batterie sont *fréquents* dans les RCSF. Le service de diagnostic identifie ces problèmes en présentant plusieurs caractéristiques qui favorisent notre approche IFTF. Le service est générique. Il fournit la tâche de diagnostic à de nombreux scenarios de RCSF (déploiement dispersé ou dense, application de collecte de données ou à base d'événements, *etc*.). D'un point de vue conception, le service est implémenté dans une couche indépendante de la pile protocolaire, mais facilement intégrée avec les couches de la pile. Il fournit de nombreux paramètres ajustables en fonction des besoins de l'application. Le service est flexible. Il s'adapte aux changements topologiques, en gérant le nombre de voisins surveillé par chaque nœud d'une manière dynamique. En outre, le service est économe en énergie et permet le passage à l'échelle. Il maîtrise la dissipation d'énergie grâce principalement aux mécanismes de gestion des voisins et à la politique de gestion des temporisateurs. Les résultats expérimentaux montrent que le service, est léger en terme de consommation des ressources, tout en fournissant un taux de détection et une précision de diagnostic satisfaisants. Grâce au service de diagnostic, IFTF peut fournir un compte-rendu concernant les éléments qui tombent en panne afin de déclencher les opérations de maintenance nécessaires.

6.2.3. Test d'applications

Le service de test est le composant cœur de l'architecture IFTF. Il représente une contribution importante de cette thèse. En effet, les approches de tests en ligne présentées dans la littérature de RCSF sont limitées. La majorité de ces approches testent l'application hors ligne ou d'une manière post mortem. Le service se caractérise par la capacité à détecter les fautes les plus *critiques,* liées au fonctionnement de l'ensemble du système. Le test permet vérifie si les différentes fonctionnalités offertes par l'application fonctionnent comme prévu. Le test permet l'isolation des fautes en se basant sur des règles de dépendance de test. L'avantage de l'isolation des fautes est double : elle favorise les opérations de maintenance et permet de réduire la consommation énergétique induite par le test. Grâce à ce service, IFTF assure une couverture potentielle des fautes et fournit un compte-rendu sur la validation de l'application en ligne et le fonctionnement global du système.

6.2.4. Architecture globale

Notre architecture IFTF combine les trois outils de la détection d'anomalies dans un seul Framework. Le service de diagnostic et de test résident dans chaque nœud du réseau. L'intégration des deux services s'effectue à travers un gestionnaire IFTF. Ce dernier assure l'interaction de la couche d'application avec les deux services. Le système de détection d'anomalies des données réside au niveau « clusterhead ». IFTF est global puisqu'il traite les classes de fautes qui affectent le RCSF. L'architecture

hiérarchique du réseau ainsi que les mécanismes de gestion d'énergie de deux services, test et diagnostic, permettent le *passage à l'échelle* de la solution. L'approche est *générale*, nous avons adopté une approche orientée service pour traiter l'hétérogénéité des capteurs et des applications.

6.3.Perspectives

L'étude menée tout au long de cette thèse a traité la problématique de sûreté de fonctionnement des RCSF. Cela nous a permis de dégager plusieurs perspectives de recherche :

- **Résolution des limites des approches** proposées dans la thèse : 1) nous avons montré dans le chapitre 4 les limites du service de diagnostic pour détecter les défaillances à base de modèle d'une manière efficace ainsi qu'à diagnostiquer les pannes de liens. Ainsi une étude d'autres métriques de diagnostic peut résoudre ces problèmes. 2) les valeurs des paramètres des algorithmes de détection des fautes des données ont été déterminées hors ligne. Il serait possible d'optimiser le choix des paramètres en utilisant des modèles de prédiction sophistiqués permettant de déterminer les valeurs des paramètres en ligne et en fonction du temps.

- **Optimisation de la consommation d'énergie à plusieurs niveaux** :

  1) la combinaison des paquets de test et de diagnostic dans un seul paquet,
  2) le recours aux techniques de modélisation des données, par exemple les ondelettes, permettant de réduire la quantité d'informations nécessaires à la détection/classification des fautes des données,
  3) introduction d'un mécanisme intelligent pour le déclenchement des services de test et de diagnostic en fonction de la criticité des zones géographiques, des fonctionnalités à tester et de la durée de déploiement.

- **Génération automatique des cas des tests** : les cas des tests dans notre approche peuvent être configurés avant le déploiement selon le type de l'application. Une façon de diminuer le coût de développement est de générer automatiquement les cas de tests à partir de la spécification de l'application. Dans la littérature, les travaux dans [106] [170] [172] mettent en place une génération de cas de test à partir d'une spécification d'application effectuée grâce à un langage de spécification SNEDL ou à partir des données archivées selon des techniques « Record and Replay ».

- **Enrichissement de l'approche globale par des mécanismes de reconfiguration et recouvrement** : les résultats fournis par l'approche globale peuvent être exploités pour concevoir des outils de recouvrement. Par exemple, en se basant sur la classification des fautes des données, il serait facile de corriger les données en se basant sur des modèles de prédiction à court et à long terme [117]. De même, les résultats de tests en ligne fournissent un moyen efficace pour évaluer les performances du réseau, l'adaptation de l'application à la dynamique de l'environnement. Par suite, il serait bénéfique de mettre en œuvre des outils de

reconfiguration fondés sur les résultats en ligne. Dans la littérature, les approches de reconfiguration de l'application ont été développées dans le but de s'adapter aux changements du contexte de l'exécution du système. De telles approches peuvent compléter notre approche de test et de diagnostic en ligne.

BIBLIOGRAPHIE

[1] http://www.businessweek.com/datedtoc/1999/9935.htm

[2] S. Cheung, S. Coleri, P. Varaiya. Sensor networks for monitoring traffic. The 42nd Allerton Conference on Communication, Control and Computing, October 2004, Monticello, Illinois.

[3] http://www.pods.hawaii.edu/.

[4] D. Estrin, D. Culler, K. Pister, G. Sukhatme. Connecting the physical world with pervasive networks. IEEE Pervasive Computing, March 2002, Los Angeles, USA, pp. 59 – 69.

[5] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, J. Anderson. Wireless sensor networks for habitat monitoring. The 1st ACM international workshop on Wireless sensor networks and applications, September 2002, Atlanta, USA, pp. 88 – 97.

[6] K.Ren, K. Zeng, W. Lou. Secure and Fault-Tolerant Event Boundary Detection in Wireless Sensor Networks. IEEE Transactions on Wireless Communications, January 2008, Chicagou, USA, pp. 354 – 363.

[7] T. Yuan, S.Zhang. Secure Fault Tolerance in Wireless Sensor Networks. IEEE 8th International Conference on Computer and Information Technology Workshops, July 2008, pp. 477-482.

[8] C. rodriguez. Qualité des données capteurs pour les systèmes de surveillance de phénomènes environnementaux. L'institut national des Sciences Appliquées de Lyon, thèse 2010.

[9] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam E. Cayirci. Wireless sensor networks: a survey. Computer Networks 2002, pp. 393-422.

[10] K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, G.S. Sukhatme. Robomote: Enabling Mobility in Sensor Networks. The 4th International Symposium on Information Procesing in Sensor Networks, April 2005, Los Angeles, USA, pp. 404 – 409.

[11] S. Chen, M. Coolbeth, H. Dinh, Y.-A. Kim, B. Wang. Data Collection with Multiple Sinks in Wireless Sensor Networks. The 4th International Conference on Wireless Algorithms, Systems, and Applications, August 2009, Boston, USA, pp. 284 – 294.

[12] B. Guizani. Algorithme de clusterisation et protocols de routage dans les réseaux ad hoc. Laboratoire Systèmes et Transport de l'Université de Technologie de Belfort Montbéliard, thèse 2012.

[13] J. Hill, M. Horton, R. Kling, L Krishnamurthy. The Platforms Enabling Wireless Sensor Networks. Communications of the ACM - Wireless sensor networks, June 2004, New York, USA, pp. 41-46.

[14] http://www.jlhlabs.com/jhill_cs/spec/index.htm

[15] http://www.xbow.com.

[16] Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), IEEE Std. 802.15.4. October 2003.

[17] http://www.intel.com/research/exploratory/motes.htm

[18] http://www.bluetooth.org/.
[19] W. Dong, C. Chen, X. Liu , J. Bu. Providing OS Support for Wireless Sensor Networks : Challenges and Approaches. IEEE Communications surveys and tutorials, 2010.
[20] http://fr.wikipedia.org/wiki/Système_d'exploitation_pour_capteur_en_réseau
[21] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse1, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, D. Culler. TinyOS: An operating system for sensor networks. In Ambient Intelligence, Springler, 2005, pp. 115-148.
[22] M. Farooq and T. Kunz. Operating Systems for Wireless Sensor Networks: A Survey. Sensors, May 2011, Ottawa, Canada, pp. 5900-5930.
[23] http://mantis.cs.colorado.edu/
[24] http://www.liteos.net/
[25] http://www.contiki-os.org
[26] L. Saraswat, P. Yadav. A Comparative Analysis of Wireless Sensor Network Operating Systems. International Journal of Engineering and Technoscience, October 2010, Ghaziabad, India, pp. 41-47.
[27] L. Yu, N.Wang, X. Meng. Real-time forest fire detection with wireless sensor networks. International Conference on Wireless Communications, Networking and Mobile Computing, September 2005, Shanghai, China, pp. 1214 – 1217.
[28] G. Werner-Allen, J. Johnson, M. Ruiz. J. Lees, M. Welsh. Monitoring volcanic eruptions with a wireless sensor network. Second European Workshop on Wireless Sensor Networks, February 2005, Boston, USA, pp. 108 – 120.
[29] D. Hughes, P. Greenwood, G. Coulson, G. Blair, F. Pappenberger, P.Smith, K. Beven. An Intelligent and Adaptable Flood Monitoring and Warning System. The 5th UK E-Science All Hands Meeting, September 2006, Nottingham, UK.
[30] K. Lorincz, D. Malan, T. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, M. Welsh, S.Moulton. Sensor networks for emergency response: Challenges and opportunities. IEEE Pervasive Computing, October 2004, New Jersey, USA, pp. 16-23.
[31] H. Furtado, R. Trobec. Applications of wireless sensors in medicine. The 34th International Convention, May 2011, Vienna, Austria, pp. 257 – 261.
[32] A. Bonivento, L. Carloni, A. Sangiovanni-Vincentelli. Platform-based design of wireless sensor networks for industrial applications. The conference on Design, automation and test in Europe, March 2006, Berkeley, USA, pp. 1 – 6.
[33] C.Chong, S. Kumar. Sensor networks: evolution, opportunities, and challenges. The IEEE Proceedings, August 2003, pp. 1247 – 1256.
[34] A.Tariq, M. Hammoudeh, Z. Bandar, A. Nisbet. An Overview and Classification of Approaches to Information Extraction in Wireless Sensor Networks. The 5th International Conference on Sensor Technologies and Applications, August 2011, Nice, France, pp. 255-260.
[35] B. Gedik, L. Liu, and P. S. Yu. Asap: An adaptive sampling approach to data collection in sensor networks. IEEE Transactions on Parallel and Distributed Systems, December 2007, NJ, USA, pp. 1766-1783.
[36] C. Zhang, C. Wang, D. Li, X. Zhou, C. Gao. Unspecific event detection in wireless sensor networks. International Conference, Communication Software and Networks 2009, Shanghai, China, pp. 243 – 246.

[37] C. Chien-Chung Shen. Srisathapornphat, C. Jaikaeo. Sensor information networking architecture and applications. Personal Communications, August 2001, New Jersey, USA, pp. 52 – 59.
[38] D. Curiac, C.Volosens. Redundancy and Its Applications in Wireless Sensor Networks: A Survey**.** Transactions on computers, April 2009, New York, USA, pp.705-714
[39] A.Taleb, D. K. Pradhan, T.Kocak. A Technique to Identify and Substitute Faulty Nodes in Wireless Sensor Networks. Third International Conference on Sensor Technologies and Applications, June 2009, Bristol, UK, pp. 346 - 351.
[40] Saleh, A. Agbaria, M. Eltoweissy. In-Network Fault Tolerance in Networked Sensor Systems. 2nd ACM/SIGMOBILE Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks, September 2006, California, USA, pp. 47 – 54.
[41] F. Koushanfar, M. Potkonjak, A.Vincentelli. Fault Tolerance in Wireless Ad-Hoc Sensor Networks. IEEE Sensors, June 2002, p. 1491-1496.
[42] http://tice.utc.fr/moodle/file.php/498/SupportWeb/co/Module_RCSF_53.html
[43] Boukerche, R. Werner, N. Pazzi. A Fast and Reliable Protocol for Wireless Sensor Networks in Critical Conditions Monitoring Applications. 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems, October 2004, New York, USA, pp. 157-164.
[44] F. Benhamida, Y. Challal. FaT2D: Fault Tolerant Directed Diffusion Protocol. International Conference on Availability, Reliability and Security, February 2010, Algiers, Algeria, pp. 112 - 118.
[45] C. Intanagonwiwat, R. Govindan D.Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. 6th annual international conference on Mobile computing and networking, August 2000, New Yor, USA, p. 56–67.
[46] Lazaridis, Q. Han, S. Mehrotra, N. Venkatasubramanian. Fault Tolerant Evaluation of Continuous Selection Queries over Sensor Data. International Journal of Distributed Sensor Networks, July 2009, Bristol USA, pp. 338–360.
[47] E. Rahm H. Do. Data cleaning: Problems and current approaches. IEEE Bulletin of the Technical Committee on Data Engineering, December 2000, Leipzig, Germany, pp. 3-13.
[48] E. Elnahrawy, B. Nath. Cleaning and querying noisy sensors. 2nd ACM international conference on Wireless sensor networks and applications, September 2003, New York, USA, pp. 78 – 87.
[49] S.Jeffery, G. Alonso, M. Franklin, W. Hong, J. Widom. A pipelined framework for online cleaning of sensor data streams. 22nd International Conference on Data Engineering, April 2006, Berkeley, USA, pp. 140.
[50] F. Koushanfar, M. Potkonjak A. Vincentelli. Online fault detection of sensor measurements. IEEE Sensors, October 2003, Berkeley, USA, pp. 974 – 979.
[51] J. Chen, S. Kher, A. Somani. Distributed fault detection of wireless sensor networks.Workshop on Dependability issues in wireless ad hoc networks and sensor networks, 2006, New York, USA, pp. 65 – 72.
[52] L. Myeong-Hyeon, C. Yoon-Hwa. Distributed diagnosis of wireless sensor networks. IEEE Region 10 Conference, November 2007, Hongik, Seoul pp. 1 – 4.

[53] X. Xiang-hua, Z. Biao, W. Jian. Tree Topology based Fault Diagnosis in Wireless Sensor Networks. International Conference on Wireless Networks and Information Systems, December 2009, Hangzhou, China, pp. 65 - 69.

[54] M. Ding, D. Chen, K. Xing, X. Cheng. Localized fault-tolerant event boundary detection in sensor networks. 24th Annual Joint Conference of the IEEE omputer and Communications Societies, March 2005, Washington, USA, pp. 902 – 913.

[55] N. Ramanathan, K. Chang, L. Girod, R. Kapur, E. Kohler D. Estrin. Sympathy for the sensor network debugger. 3rd international conference on Embedded networked sensor systems, May 2005, New York, USA, pp. 255 - 267.

[56] X. Miao, K. Liu, Y. He, Y.Liu, D. Papadias. Agnostic Diagnosis: Discovering Silent Failures in Wireless Sensor Networks. IEEE INFOCOM, April 2011, Shanghai, China, pp. 1548 – 1556.

[57] L. Mo, Y. He, Y. Liu. Canopy Closure Estimates with GreenOrbs: Sustainable Sensing in the Forest. 7th ACM Conference on Embedded Networked Sensor Systems, November 2009, Berkeley, USA, pp. 99-112.

[58] B. Chen, G. Peterson, G. Mainland, M. Welsh. LiveNet: Using Passive Monitoring to Reconstruct Sensor Network Dynamics. 4th IEEE international conference on Distributed Computing in Sensor Systems, April 2008, Heidelberg, Berlin, pp. 79-98.

[59] M. Ringwald, K. Romer, A. Vitaletti. SNTS: Sensor Network troubleshooting Suite.3rd IEEE international conference on Distributed computing in sensor systems, June 2007, Heidelberg, Berlin, pp. 142-157.

[60] K. Whitehouse, G. Tolle, J. Taneja, C. Sharp, S. Kim, J. Jeong, J. Hui, P. Dutta, D. Culler. Marionette: Using RPC for interactive development and debugging of wireless embedded networks. Fifth International Conference on Information Processing in Sensor Networks, Berkeley, USA, April 2006, pp. 416 – 423.

[61] J. Yang, M. L. Soffa, L. Selavo, K. Whitehouse. Clairvoyant: A comprehensive source-level debugger for wireless sensor networks. 5th international conference on Embedded networked sensor systems, November 2007, Sydney, Australia, pp. 189-203.

[62] http://www.hkcloud.net/Sengraphy.

[63] Y. Zhou, X. Chen, M. Lyu, and J. Liu. Sentomist: Unveiling transient sensor network bugs via symptom mining. 30th International Conference on Distributed Computing Systems. Genoa, Italy, June 2010, Genoa, Italy, pp. 784–794.

[64] M. Lodder, G.P. Halkes, K.G. Langendoen. A global-state perspective on sensor network debugging. 5th ACM Workshop on Embedded Networked Sensors, June 2008, Virginia, USA.

[65] A.Tavakoli, D.Culler, S. Shenker.The case for predicate-oriented debugging of sensornets. 5th ACM Workshop on Embedded Networked Sensors, June 2008, Virginia, USA.

[66] M. Khan, H. Le, M. May, P. Moinzadeh, L. Wang, Y. Yang, D. Noh, T. Abdelzaher, C. Gunter, J Han, X. Jin. Diagnostic powertracing for sensor node failure analysis. 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, April 2010 Stockholm, Sweden, pp. 117-128

[67] A. Schoofs, G. Hare, A. Ruzzelli. Debugging Low-Power and Lossy Wireless Networks: A Survey. IEEE Communications Surveys and Tutorials, Second Quarter 2011, Dublin, Ireland, pp. 311 - 321

[68] G.Tolle, D. Culler. Design of an application-cooperative management system for wireless sensornetworks. Second European Workshop on Wireless Sensor Networks, February 2005, Istanbul, Turkey, pp. 121-132.

[69] A. de Jong, M. Woehrle, K. Langendoen. MoMi - Model-Based Diagnosis Middleware for Sensor Networks. 4th international workshop on Middleware Tools, Services and Run-Time Support for Sensor Networks, December 2009, New York, USA, pp. 19-24.

[70] M. Ringwald, K. Römer and A. Vitaletti. Snif: Sensor network inspection framework. Technical Report 535, Institute for Pervasive Computing, October 2006, ETH Zurich, Switzerland.

[71] http://www.memoireonline.com/08/10/3831/m_Approche-distribuee-pour-la-securite-dun-reseau-de-capteurs-sans-fils-RCSF1.html

[72] F. Dressler, R. Nebel, A. Awad. Distributed Passive Monitoring in Sensor Networks. 26th IEEE Conference on Computer Communications, 2007, Anchorage, USA.

[73] L. Ruiz, J. Nogueira, A. Loureiro. MANNA: management architecture for wireless sensor networks. IEEE Communications Magazine, February 2003, Gerais, Brazil, pp. 116 – 125.

[74] K. Liu, Q. Ma, X. Zhao, and Y. Liu. Self-diagnosis for large scale wireless sensor networks. IEEE INFOCOM, April 2011 Shanghai, China, p. 1539 – 1547.

[75] M. Yu, H. Mokhtar, M. Merabti. A survey on Fault Management in wireless sensor network. 8th Annual Post Graduate Symposium on The Convergence of Telecommunications, Networking and Broadcasting, June 200, New York, USA, pp. 171 – 190.

[76] R. Jurdak, X. Rosalind Wang, O. Obst, and P.Valencia.Wireless Sensor Network Anomalies: Diagnosis and Detection Strategies. Chapter book, Intelligence-based Systems Engineering, Andreas Tolk and Lakhmi Jain, editors, Springer, 2011, pp. 308- 25.

[77] A. Rodrigues, T. Camilo, J. Silva, F. Boavida. Diagnostic Tools for Wireless Sensor Networks: A Comparative Survey. Journal of Network and Systems Management, Springer US, June 2012.

[78] K. Liu, M. Li, Y. Liu, M. Li, Z. Guo, and F. Hong. Passive Diagnosis for Wireless Sensor Networks. IEEE/ACM Transactions on Networking, August 2010, Kowloon, China, p. 1132 – 1144.

[79] M. M. H. Khan, H. K. Le, H. Ahmadi, T. F. Abdelzaher, and J. Han. Dustminer: Troubleshooting interactive complexity bugs in sensor networks. 6th ACM conference on embedded network sensor systems, November 2008, New York, USA, pp. 99–113.

[80] http://www.cse.ust.hk/~liu/Ocean/index.html

[81] Z. Yang, M. Li, and Y. Liu. Sea depth measurement with restricted floating sensors. 28th IEEE International Real-Time Systems Symposium, December 2007, Washington, USA, pp. 469–478.

[82] Z. Chen, K. Shin. Post-deployment performance debugging in wireless sensor networks. 30th IEEE Real-Time Systems Symposium, December 2009, Michigan, USA, pp. 313 – 322.
[83] S. Salim, M. Javed and A. Akkbar. A mobile agent-based Architecture for Fault tolerance in Wireless sensor Networks. 8th Annual Communication Networks and Services Research Conference, May 2010, Quebec, Canada, pp. 276 – 283.
[84] H. Wedde, M. Farooq, T. Pannenbaecker, B.Vogel. BeeAdHoc: Energy Efficient Routing Algorithm for Mobile Ad Hoc Networks Inspired by Bee Behavior. Conference on Genetic and evolutionary computation, June 2005, New York, USA, pp. 153-160.
[85] G. Gupta and M. Younis. Fault-Tolerant Clustering of Wireless Sensor Networks. IEEE Wireless communications and Networking, March 2003, Baltimore, USA, pp.1579–1584.
[86] S. Rost, H. Balakrishnan. Memento: a health monitoring system for wireless sensor networks. 3rd IEEE Annual Communications Society on Sensor and Ad Hoc Communications and Networks, September 2006, Reston, USA, pp. 75- 584.
[87] X. Liu, Z. Guo, X. Wang, F. Chen, X. Lian, J. Tang, M. Wu, M. F. Kaashoek, Z. Zhang. D3S: Debugging deployed distributed systems. 5th USENIX Symposium on Networked Systems Design & Implementation, April 2008, California, USA, pp. 423-437.
[88] D. Geels, G. Altekar, P. Maniatis, T. Roscoe. Friday: Global comprehension for distributed replay. 5th USENIX Symposium on Networked Systems Design & Implementation, April 2007, Cambridge, England, p. 285–298.
[89] G. Barrenetxea, F. Ingelrest, G. Schaefer, M. Vetterli. The Hitchhiker's Guide to Successful Wireless Sensor Network Deployments. 6th ACM conference on embedded network sensor systems, November 2008, New York, USA, pp. 43-56.
[90] K. Langendoen, A. Baggio, O. Visser.Murphy Loves Potatoes: Experiences from a Pilot Sensor Network Deployment in Precision Agriculture. International Parallel and Distributed processing symposium, April 2006, Rhodes Island, USA, pp. 8.
[91] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, W. Hong. A Macroscope in the Redwoods. 3rd international conference on Embedded networked sensor systems, November 2005, San Diego, USA, pp. 51 – 63.
[92] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, M. Welsh. Fidelity and Yield in a Volcano Monitoring Sensor Network. 7th symposium on Operating systems design and implementation, November 2006, Seattle, USA, pp. 381-396.
[93] P. Levis, N. Lee, M. Welsh, D. Culler. TOSSIM: accurate and scalable simulation of entire tiny OS applications. 1st international conference on Embedded networked sensor systems, November 2003, California, USA, pp. 126 – 137.
[94] F. Osterlind, A.Dunkels, J.Eriksson, N. Finne, T.Voigt. Cross-level sensor network simulation with COOJA. IEEE Conference on Local Computer Networks, November 2006, Florida, USA, pp. 641 – 648.
[95] http://www.omnetpp.org
[96] http://nsnam.isi.edu/nsnam/index.php/Main Page
[97] http://pcl.cs.ucla.edu/projects/glomosim/

[98] http://wsnet.gforge.inria.fr/index.html
[99] A. Sobeih, W. P. Chen, J. C. Hou, L. C. Kung, N. Li, H. Lim, H. Y. Tyan, H. Zhang. J-Sim: A Simulation Environment for Wireless Sensor Networks. 38th Annual Simulation Symposium, April 2005, Urbana, USA, pp. 175 – 187.
[100] R. Sasnauskas, O. Landsiedel, M.H. Alizai, C. Weise, S. Kowalewski, K. Wehrle. KleeNet: Discovering Insidious Interaction Bugs in Wireless Sensor Networks Before Deployment. 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, April 2010, Stockholm, Sweden, pp. 186-196.
[101] P. Li, J. Regehr. T-check: bug finding for sensor networks. 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, April 2010, Stockholm, Sweden, pp. 174-185.
[102] A. Wichmann. Inter-Context Control-Flow Graph for NesC, with Improved Split-Phase Handling. 16th ACM International Symposium on Foundations of software engineering, July 2010, New York, USA, pp. 94-104.
[103] V. Krunic, E. Trumpler, R. Han. NodeMD: diagnosing node-level faults in remote wireless sensor systems. 5th international conference on Mobile systems, applications and services, June 2007, San Juan, Puerto Rico, pp. 43-56.
[104] K. Romer. Passive distributed assertions for sensor networks. 2009 International Conference on Information Processing in Sensor Networks, April 2009, San Francisco, USA, pp. 337-348.
[105] D. Herbert, Y. Lu, S. Bagchi and Z. Li. Detection and Repair of Software Errors in Hierarchical Sensor Networks. IEEE International Conference on Sensor Networks, Ubiquitous and Trustworthy Computing, June 2006, Taichung, Taiwan, pp. 403 – 410.
[106] Y. Wu, K. Kapitanova, J. Li, J. Stankovic, S. Son and K. Whitehouse. Run Time Assurance of Application-Level Requirements in Wireless Sensor Networks. 9th ACM/IEEE Conference on Information Processing in Sensor Networks, April 2010, Stockholm, Sweden, pp. 197-208.
[107] B. Jiao. Snedl: Sensor network event description language for event service architecture. Master's thesis, Department of Computer Science, University of Virginia, 2005.
[108] M. Kumar, L. Schwiebert M. Brockmeyer. Efficient data aggregation middleware for wireless sensor networks. 1st International Conference on Mobile, Ad-Hoc, and Sensor Systems, October 2004 Florida, USA, pp.579-581.
[109] E. Cayirci. Data Aggregation and Dilution by modulus addressing in wireless sensor networks. IEEE Communications Letters, August 2003, Istanbul, Turkey, pp. 355–357.
[110] M. Anisi, J. Rezazadeh, M. Dehghan. FEDA: Fault-tolerant Energy-Efficient Data Aggregation in Wireless Sensor Networks. 16th International Software, Telecommunications and Computer Networks, September 2008, Split, Croatia, pp. 188 – 192.
[111] Y. Yao, J. Gehrke. The cougar approach to in-network query processing in sensor networks. International Conference on Management of Data, Wisconsin, USA, September 2002, pp. 9–18.
[112] Y.Zhenyu, Z. Hai, L. Kai, S. Peigang, G.Yishan, Z. Yongqing, X. Ye. Multi-sensor Data fusion in wireless sensor networks. Multiconference on

Computational Engineering in Systems Applications, October 2006, Beijing, China, pp.1690 – 1694.
[113] Boukerche, A. Martirosyan. An Energy-Aware and Fault Tolerant Inter-Cluster communication Based Protocol for Wireless Sensor Networks. IEEE Global Telecommunications Conference, November 2007, Washington, USA, pp. 1164 – 1168.
[114] L. Gürgen. Gestion à grande échelle de données de capteurs hétérogènes. Institut National Polytechnique de Grenoble, 2007.
[115] L. Gürgen, J. Persson, A. Cherbal, C. Labbé, C. Roncancio, S. Honiden. Plug&manage heterogeneous sensing devices. 6th International Workshop on Data Management for Sensor Networks, August 2009, Lyon, France.
[116] M. Faulkner, M. Olson, R. Chandy, J. Krause. The Next Big One: Detecting Earthquakes and other Rare Events from Community-based Sensors. 10th International Conference on Information Processing in Sensor Networks, April 2011, Chicago, USA, pp. 13-24.
[117] R. Zhu. Efficient Fault-Tolerant Event Query Algorithm in Distributed Wireless Sensor Networks. International Journal of Distributed Sensor Networks, 2010.
[118] S. Zahedi, M. Szczodrak, P. Ji, D. Mylaraswamy, M. Srivastava, R. Young. Tiered architecture for on-line detection, isolation and repair of faults in wireless sensor networks. IEEE Military Communications Conference, November 2008, San Diego, CA, pp. 1-7.
[119] Y.Zhang, N. Meratnia, Paul Havinga. Adaptive and Online One-Class Support Vector Machine-based Outlier Detection Techniques for Wireless Sensor Networks. International Conference on Advanced Information Networking and Applications Workshops. May 2009, Washington, USA, pp. 990-995.
[120] T. Lim. Detecting anomalies in Wireless Sensor Networks. Qualifying Dissertation. University of York. August 2010.
[121] A.Singh, B. Giridhar, P. Mandal. Fixing Data Anomalies with Prediction Based Algorithm in Wireless Sensor Networks. 7th IEEE International Conference on Wireless Communication and Sensor Networks, December 2011, Panna, India.
[122] A. Ghaddar, Tahiry Razafindralambo, Isabelle Simplot-Ryl, Samar Tawbi, Abbas Hijazi. Algorithm for temporal anomaly detection in WSNs. IEEE Wireless Communication and Networking Conference, March 2011, Quintana-Rio, Mexico, pp. 743-748.
[123] I. Onat and A. Miri. An intrusion detection system for wireless sensor networks. IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, August 2005, pp. 253-259.
[124] http://db.csail.mit.edu/labdata/labdata.html.
[125] S. Madden, M. Franklin, J. Hellerstein, W. Hong. Tinydb: An acquisitional query processing system for sensor networks. Journal ACM Transactions on Database Systems, March 2005, New York, USA, pp. 122-173.
[126] K. Ni, N. Ramanathan , M.Chehade , L.Balzano , S.Nair , S. Zahedi , E. Kohler , G. Pottie , M. Hansen , M. Srivastava. Sensor network data fault types. ACM Transactions on Sensor Networks, May 2009, New York, USA.
[127] V. Hodge and J. Austin. A survey of outlier detection methodologies. Artificial Intelligence Review, 2004, pp. 85-126.

[128] A.Sharma, L.Gollubich, R. Govindan. Sensor Faults: Detection Methods and Prevalence in Real-World Datasets. ACM Transactions on Sensor Networks, June 2010, New York, USA.
[129] N. Ramanathan, L.Balzano, M. Burt, D. Estrin, E. Kohler, T. Harmon, C. Harvey, Jay, S. Rothenberg, M. Srivastana. Rapid Deployment with Confidence: Calibration and Fault Detection in Environmental Sensor Networks. CENS Tech Report #62, April 2006.
[130] H. Karl, A. Willig. A short survey of wireless sensor networks. Technical Report, University Berlin - Telecommunication Networks Group, October 2003.
[131] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. Neurocomputing. Foundations of Research, MIT Press, 1988.
[132] U. Alfred, H. Peter. Kohonen's Self Organizing Feature Maps for Exploratory Data Analysis. In Widrow, Bernard; Angeniol, Bernard. International Neural Network Conference, July 1990, Paris, France.
[133] M. Beale, M. Hagan, H. Demuth. Neural Network Toolbox. User's guide. March 2012.
[134] Wilppu, E. The visualization capability of self-organizing maps to detect deviations in distribution control, Technical Report 153, Turku Centre for Computer Science. 1997.
[135] G. Barreto, J. Mota, L. Souza, R. Frota, L. Aguaya. Condition monitoring of 3G cellular network through, IEEE Trans. Neural Networks, September 2006, pp. 1064-1075.
[136] P. Sukkhawatchani W. Usaha. Performance Evaluation of Anomaly Detection in Cellular Core Networks using Self-Organizing. ECTI-CON, May 2008, pp.361-364.
[137] J. Zheng, M. Hu. Detection of TCP Attacks Using SOM with Fast Nearest-Neighbor Search. International Conference on Neural Networks, 2005, pp.176-182.
[138] L. Paladina, M. Paone, G. Jellamo, and A. Puliafito. Self organizing maps for distributed localization in wireless sensor networks. 12th IEEE Symposiump on Computers and Communications July 2007, Aveiro, Portugal, pp.1113-1118,
[139] L. Souza, H. Vogt and M. Beigl. A survey on fault tolerance in wireless sensor networks. SAP Research. Karlsruhe University, 2007, Germany.
[140] M. Ringwald, K. Romera, A. Vitaletti. Passive inspection of sensor networks. 3rd IEEE international conference on Distributed computing in sensor systems, Heidelberg, Berlin, June 2007, pp. 205-222.
[141] C. Hsin, M. Liu. Self-monitoring of wireless sensor networks. Computer Communications, February 2006, Amsterdam, the Netherlands, pp. 462-476.
[142] A. Meier, M. Motani, S. Hu, K. Simon. DiMo: Distributed Node Monitoring in Wireless Sensor Networks. 11th International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems, October 2008, British Colombia, Canada, pp. 117-121.
[143] M. Asim, H.Mokhtar, M. Merabti. A self-managing fault management mechanism for wireless sensor networks. International Journal of Wireless & Mobile Networks, November 2010.

[144] G. Hackmann, O. Chipara and C. Lu. Robust Topology Control for Indoor Wireless Sensor Networks. 6th ACM conference on Embedded network sensor systems, November 2008, North Carolina, USA, pp. 57-70.
[145] O. Chipara, G. Hackmann, C. Lu, W. Smart G.C. Roman. Practical Modeling and Prediction of Radio Coverage of Indoor Sensor Networks. 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, April 2010, Stockholm, Sweden, pp. 339-349.
[146] X. Mao, X. Li, X. Shen, F. Chen. iLight: device-free passive tracking by wireless sensor networks». 7th International Conference on Embedded networked sensor systems, November 2009, California, USA, pp. 315–316.
[147] X. Mao, S. Tang, X. Li Y. Sun. MENs: Multi-user Emergency Navigation System Using Wireless Sensor Networks. Ad Hoc & Sensor Wireless Networks, 2011, pp. 23-53.
[148] A. Sheth, C. Hartung, R. Han. A Decentralized Fault Diagnosis System for Wireless Sensor Networks. 2nd IEE International Conference on Mobile Ad Hoc and Sensor Systems, September 2005, Washington, USA, pp. 194-197.
[149] S. Lai. Duty-Cycled Wireless Sensor Networks: Wakeup Scheduling, Routing, and Broadcasting, Virginia Polytechnic Institute and State University, thesis, 2010.
[150] R. Kacimi. Techniques de conservation d'énergie pour les réseaux de capteurs sans fils, Institut National Polytechnique de Toulouse, thèse, 2009.
[151] A. Salhieh. efficient communication in stationary wireless sensor networks», thesis, Wayne State University, Detroit, Michigan, 2004.
[152] I. Urteaga, K. Barnhart, and Q. Han. Redflag a run-time distributed flexible, lightweight, and generic fault detection service for data-driven wireless sensor applications. Journal of Pervasive and Mobile Computing, October 2009, Amsterdam, the Netherlands, pp. 432-446.
[153] S. Park, A. Savvides, and M. Srivastava. SensorSim: A simulation framework for sensor networks. 3rd ACM Int. Workshop on Modeling, Analysis & Simulation of Wireless and Mobile Systems, August 2000. Boston, USA, pp. 104-111.
[154] S. Dhurandher, S. Misra, M. Obaidat, S. Khairwal. UWSim: A simulator for underwater sensor networks. Simulation, July 2008, pp. 327-338.
[155] L.Titzer, D. Lee, J. Palsberg. Avrora: Scalable sensor network simulation with precise timing. 4th International conference on Information Processing in Sensor Networks, April 2005, California, USA, pp. 477 – 482.
[156] http://nescc.sourceforge.net/.
[157] http://zomobo.net/la-tinyos.
[158] Monica, A. Sharma. Comparative Study of Energy Consumption for Wireless Sensor Networks based on Random and Grid Deployment Strategies. International Journal of Computer Applications, September 2010, pp. 28–35.
[159] H. Karl, A. Willis. Protocols and Architectures for Wireless Sensor Networks. John Wiley and Sons, 2005.
[160] J. Polastre, J. Hill, D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks 2nd international conference on Embedded networked sensor systems, November 2004, Baltimore, Maryland, pp. 95-107.
[161] http://www.cs.virginia.edu/~yw5s/Run time assurance/FEDL

[162] S. Li, Y. Lin, S. Son, J. Stankovic. Event detection services using data service middleware in distributed sensor networks. 2nd international conference on Information processing in sensor networks, April 2003, California, USA, pp. 502-517.
[163] A. Meier. Safety-Critical Wireless Sensor Networks, thesis, 2009.
[164] A. Taherkordi, R. Rouvoy, Q. Le-Trung, F. Eliassen. A Self-Adaptive Context Processing Framework for Wireless Sensor Networks. 3rd international workshop on Middleware for sensor networks, December 2008, Leuven, Belgium, pp. 7-12.
[165] I. Akyildiz, I. Kasimoglu. Wireless sensor and actor networks: Research challenges. Ad Hoc Networks, October 2004, Atlanta, USA, pp. 351–367.
[166] D. Puccinelli, M. Haenggi. Wireless sensor networks: applications and challenges of ubiquitous sensing. IEEE Circuits and Systems Magazine, September 2005, pp. 19 – 31.
[167] D. Suliman, B. Paech, L. Borner. The MORABIT approach to runtime component testing. 30th Annual International Computer Software and Applications Conference, September 2006, Chicago, USA, pp. 171-176
[168] T. Bui. Service de diagnostic en ligne pour les applications à base de composants logiciels, Institut Polytechnique de Grenoble, thèse, 2009.
[170] T. Clouser, R. Thomas, M. Nesterenko. Emuli: Emulated Stimuli for Wireless Sensor Network Experimentation. Technical Report TR-KSU-CS-2007-04, Kent State University, 2007.
[171] http://www.ece.cmu.edu/~webk/sensor_networks/pub/ipsn05_hilt.pdf.
[172] L. Luo, T. He, G. Zhou, L. Gu, T.F. Abdelzaher, J.A. Stakovic. Achieving Repeatability of Asynchronous Events in Wireless Sensor Networks with EnviroLog. 25th Annual Joint Conference of the IEEE Computer and Communications Societies, April 2006, Barcelona, Spain, pp. 1 – 14.