Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen

Faculté desSciences
Département d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Master en Informatique

Option: Réseaux et Systèmes Distribués (R.S.D)

Thème

Mise en place d'un schéma de routage pour la tolérance aux pannes dans les RCSF

Réalisé par :

- **HADJ ADDA Asmaa**
- **BENALLAL Wafaa**

Présenté le 25 Juin 2014 devant le jury composé de:

- *Mme LABRAOUI Nabila* *(Président)*
- *Mr LEHSAINI Mohamed* *(Encadreur)*
- *Mr BENMAMMAR Badr* *(Examinateur)*
- *Mr BELHOUCINE Amin* *(Examinateur)*

Année universitaire: 2013-2014

Dédicaces

Remerciements

Sommaire

Liste des figures

Liste des tableaux

Introduction générale

Introduction générale

La communication sans fil et la miniaturisation des équipements électroniques ont connu un grand succès dans plusieurs domaines à l'instar des domaines scientifiques et industriels. Les avantages de ces technologies ont leurs permis d'être utilisées dans les architectures réseaux actuelles en particulier dans les réseaux de capteurs sans fil (RCSF). Ce type de réseaux résulte d'une fusion de deux disciplines de l'informatique moderne : les systèmes embarqués et les communications sans fil.

Un réseau de capteurs sans fil (RCSF) est composé généralement d'un grand nombre d'unités, appelées "motes", capables de s'auto-organiser et communiquent via des liens sans fil. Le but général d'un RCSF est la collecte d'un ensemble de paramètres de l'environnement entourant les motes, telles que la température ou l'humidité et les acheminer vers des points de collecte distants appelés puits ou stations de base.

La plupart des protocoles conçus pour les RCSF en particulier les protocoles de routage utilisent souvent le modèle du disque unitaire pour modéliser les communications radio entre les nœuds. Dans ce modèle, il est supposé que si la distance séparant deux nœuds est inférieure ou égale à la portée de communication (R_c) alors la probabilité de recevoir le message sans erreur est égale à 1. Cependant, ce modèle ne peut pas être considéré comme un modèle réaliste puisqu'il suppose que les communications sont toujours fiables et que les messages sont toujours reçus sans erreur tant que la distance euclidienne séparant un nœud émetteur u d'un nœud récepteur v est inférieure ou égale au rayon de transmission $R_c(d(u,v) \leq R_c)$. Donc, ce modèle ne prend pas en considération les fluctuations du signal radio causées par la présence de bruit, la distance séparant les deux nœuds communicants, les interférences, etc. Or, ces dernières peuvent avoir un impact significatif sur les transmissions à cause des erreurs qu'elles injectent dans les messages échangés entre les nœuds. De ce fait, il est judicieux de concevoir les protocoles dans un environnement réaliste tel que le modèle lognormal [1]. En outre, les capteurs sont sujets à des pannes et dans certains cas il y aurait des capteurs défaillants qui sont impliqués dans l'acheminement des données à la station de base. Il résulte à cet effet que les données ne peuvent arriver correctement à la station de base. Pour remédier à cette anomalie, plusieurs protocoles de routage tolérants aux pannes ont été proposés dans la littérature.

Dans ce contexte, nous proposons d'évaluer certains schémas de routage dédiés aux RCSF dans deux contextes : le premier est idéal et le deuxième est probabiliste. Dans le premier nous considérons toujours que si la distance séparant les deux nœuds communicants est inférieure ou égale à la portée de communication alors le message sera reçu d'une manière intact alors que dans le deuxième nous supposons qu'un capteur pourra cesser de fonctionner avec une certaine probabilité. Puis, nous avons varié cette probabilité pour voir son impact sur les performances du schéma de routage considéré.

Ce document s'articule autour de trois chapitres. Le premier chapitre décrit les principes et les caractéristiques des réseaux de capteurs aussi que leurs domaines d'application. Dans le deuxième chapitre, nous présentons certains protocoles de routage tolérants aux pannes conçus pour les réseaux de capteurs. Dans le troisième chapitre, nous évaluons un schéma de routage basé sur le clustering tel qu'au niveau de chaque cluster il y a deux clusterheads : un qui est principal et un autre qui est adjoint. Cette évaluation se fait sur une plateforme réelle de capteurs de type TelosB. Enfin, nous concluons notre travail et nous proposons quelques perspectives.

Chapitre I

Les réseaux de capteurs sans fil

Chapitre I

Les réseaux de capteurs sans fil

I.1 Introduction

Aujourd'hui, de nombreuses applications de connexions sans fil aussi bien dans les domaines industriels que personnels requièrent des transmissions à faibles débits organisées sous forme de réseaux et impliquent des dispositifs de petite taille et à moindre coût. Ces applications sont regroupées sous le terme de réseaux de capteurs sans fil (RCSF).

Les RCSF constituent un domaine en pleine expansion possédant de nombreuses applications potentielles et ayant des contraintes relativement différentes des systèmes de communications sans fil classiques.

Dans ce chapitre, nous allons présenter les réseaux de capteurs sans fil : leurs caractéristiques, leurs architectures de communication et leurs domaines d'applications. Nous allons discuter également les principaux facteurs et contraintes qui influencent la conception des réseaux de capteurs sans fil.

I.2 Qu'est-ce qu'un capteur ?

Un capteur appelé "*Mote*" en anglais, est un petit dispositif électronique capable de mesurer une grandeur physique observée et la transformer en une grandeur utilisable, par exemple la température, la pression, l'humidité, la glycémie, etc..[2], et de la communiquer à un centre de contrôle via une station de base.

Un nœud capteur sans fil est un composant physique, de petite taille, capable d'accomplir trois tâches complémentaires :

- le relevé d'une grandeur physique
- le traitement de l'information
- la communication avec d'autres nœuds capteurs.

I.2.1 Composants d'un capteur sans fil

Un capteur est composé au minimum d'une unité de captage, d'une unité de traitement, d'une unité de transmission et d'une unité de contrôle de l'énergie [3], comme montre la figure I-1.

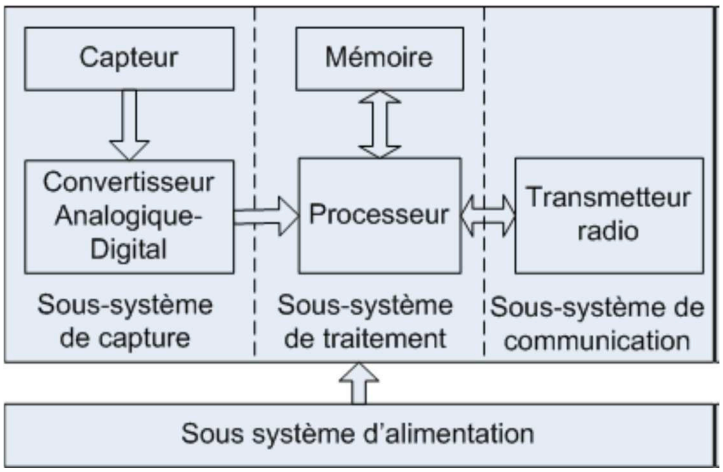

Figure I-1: Composants d'un capteur sans fil [3]

- **Unité de captage** : est constituée d'un capteur physique ou plusieurs permettant de faire des mesures sur son environnement immédiat et d'un convertisseur Analogique/Numérique (ADC[1]) qui transforme une mesure relevée en une grandeur numérique compréhensible pour la transmettre à l'unité de traitement.
- **Unité de traitement** : c'est un processeur couplé à une petite unité de stockage qui fonctionne grâce à un système d'exploitation spécifique tel que TinyOS ou Contiki. C'est cette unité qui permet le traitement de l'information selon un programme implanté par l'utilisateur.
- **Unité de transmission** : est responsable de toutes les émissions et réceptions de données via un support de communication radio.
- **Unité de contrôle de l'énergie** : elle répartit l'énergie fournie par les batteries à l'ensemble des composants actifs et met les autres en veille pour économiser de l'énergie. Cette unité est très importante car le plus grand problème des capteurs est qu'il faut économiser au maximum l'énergie pour qu'ils soient opérationnels le plus longtemps possible.

[1] ADC : Analog Digital Converter

Il existe des capteurs qui sont dotés de mobilisateur leur permettant de se déplacer et/ou de GPS pour connaitre leur position.

I.2.2 Classification des capteurs

Il existe plusieurs modes pour classifier les capteurs mais le plus répandu est celui qui tient compte de l'apport énergétique.

a) Capteurs actifs

Ce type de capteurs est constitué d'un ou plusieurs transducteurs alimentés tels que les chronomètres mécaniques. Ce sont des capteurs que l'on pourrait modéliser par des générateurs comme les systèmes photovoltaïques et électromagnétiques. D'où ils génèrent soit un courant, soit une tension en fonction de l'intensité du phénomène physique mesuré.

b) Capteurs passifs

Ces capteurs n'ont pas besoin d'apport d'énergie extérieure pour fonctionner. Ils sont modélisables par une impédance. Ainsi une variation du phénomène physique mesuré engendre une variation de l'impédance.

I.2.3 Caractéristiques des capteurs

Pour illustrer les caractéristiques matérielles des capteurs, nous présentons deux types capteurs : MicaZ et TelosB. Le tableau I-1 récapitule les caractéristiques de ces capteurs.

Tableau I-1 : Caractéristiques de MicaZ et TelosB

Caractéristiques	TelosB	MicaZ
Flash	48Kb	512Kb
RAM	10Kb	4Kb
MCU	1MHz T1 MPS430	8MHz ATmega128
Consommation d'énergie	Radio+cpu :75mW Sleep mode : 140µW	Radio+cpu :63mW Sleep mode : 30µW
External flash	512kB	512kB

I.3 Réseaux de capteurs sans fil

Un réseau de capteurs sans fil est un type particulier de réseau ad-hoc puisque la topologie du réseau est dynamique (il y a arrivée et départ de nœuds en tout moment). Il est défini par un ensemble 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 [4]. En outre, la taille du réseau est généralement très grande pour faire face aux pannes qui peuvent intervenir au niveau des capteurs. De ce fait, quand un nœud cesse de fonctionner il sera remplacé par un autre se trouvant dans sa proximité. Ceci est dans le but de garantir la fidélité de couverture de zone.

Un réseau de capteurs se compose de deux types de nœuds : des capteurs simples et des collecteurs d'informations appelés puits ou stations de base.

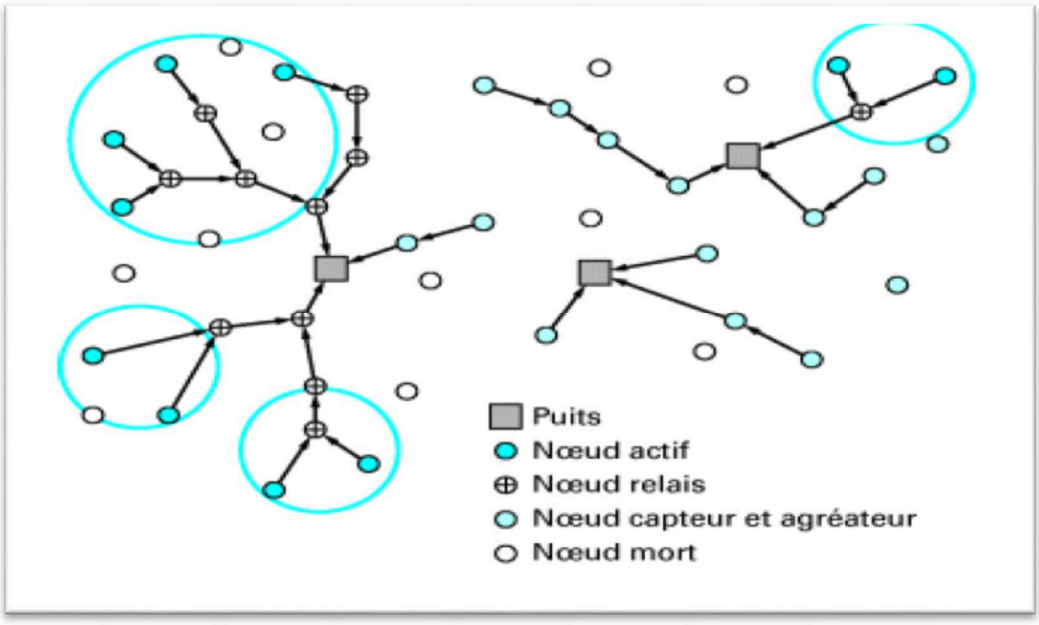

Figure I-2 : Exemples de réseaux de capteurs

I.3.1 Architecture d'un RCSF

Un RCSF est composé de capteurs généralement déployés en grand nombre et munis de transmetteurs sans fil. La figure I-4 illustre l'architecture et le fonctionnement d'un réseau de capteurs.

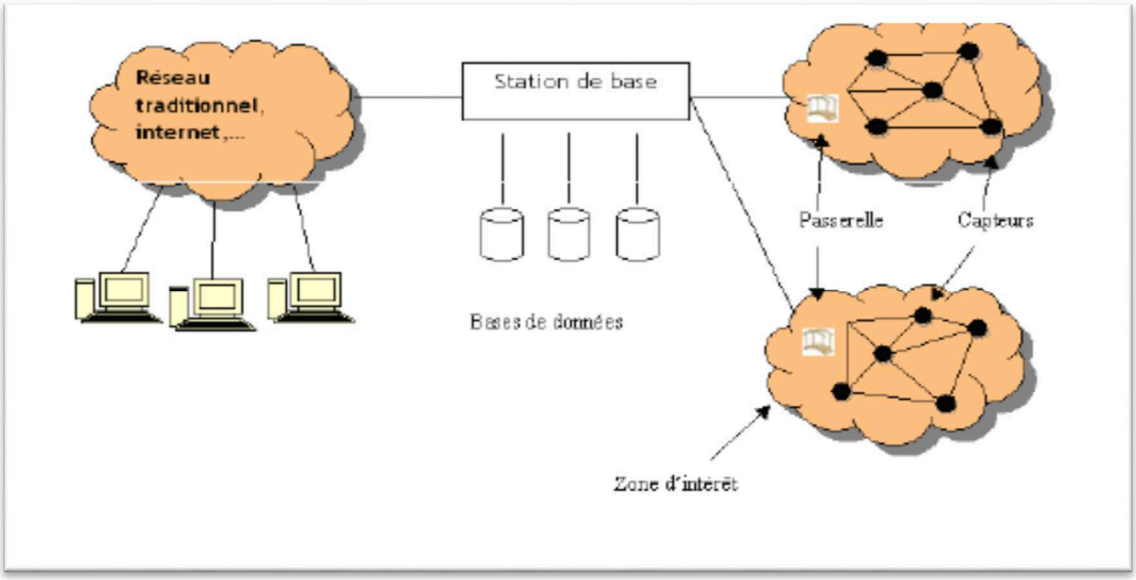

Figure I-3: Architecture et fonctionnement d'un RCSF

Dans un RCSF, les capteurs sont généralement déployés en grand nombre (par centaines ou milliers) pour garantir la tolérance aux pannes, avec une densité de nœuds généralement de l'ordre de 20 nœuds/m^3. La transmission de l'information entre capteurs peut s'effectuer à l'aide d'un support radio, infrarouge, ou optique.

Chaque capteur est susceptible de collecter des données (traitées ou pas) de l'environnement et de les acheminer vers la station de base la plus proche en un ou plusieurs sauts via des nœuds relais nommés passerelles. Ces derniers peuvent être fixes ou mobiles et transmettent ensuite ces données à centre de contrôle distant via Internet ou par satellite. Le centre de contrôle pourrait être un simple ordinateur central relié un réseau dont le rôle est l'analyse des données rapportées, et la prise de décisions comme le filtrage et l'agrégation pour en extraire l'information utile.

I.3.2 Fonctionnement des RCSF

Il y a deux méthodes pour collecter les informations dans un réseau de capteurs [5]:

a) A la demande

Lorsque nous souhaitons avoir l'état de la zone de couverture à un moment T, le nœud puits émet des brodcasts vers toute la zone pour que les capteurs remontent leur dernier relevé vers le puits. Les informations sont alors acheminées par le biais d'une communication multi-sauts comme s'est illustré par la figure I-5.

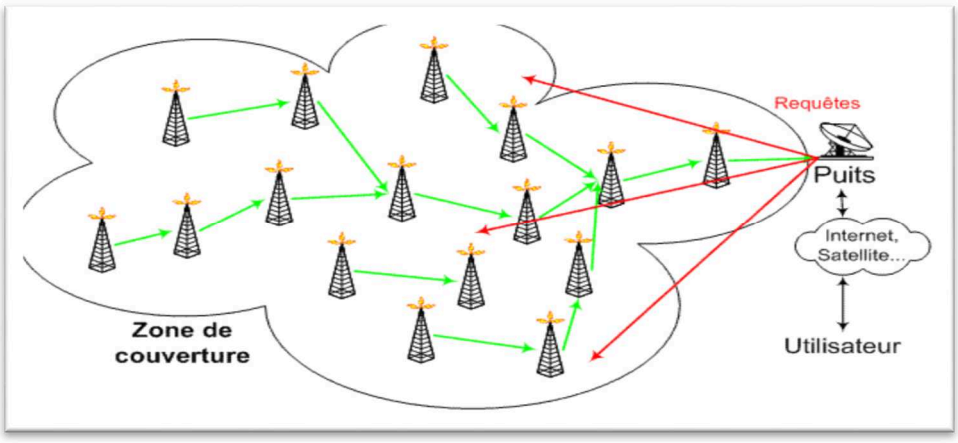

Figure I-4: Collecte d'informations à la demande

b) Suite à un événement

Ce mode de communication de l'information est utilisé lorsqu'un événement se produit en un point de la zone de couverture (changement brusque de température, mouvement...), les capteurs situés à proximité remontent alors les informations relevées et les acheminent jusqu'au puits (figure I-6).

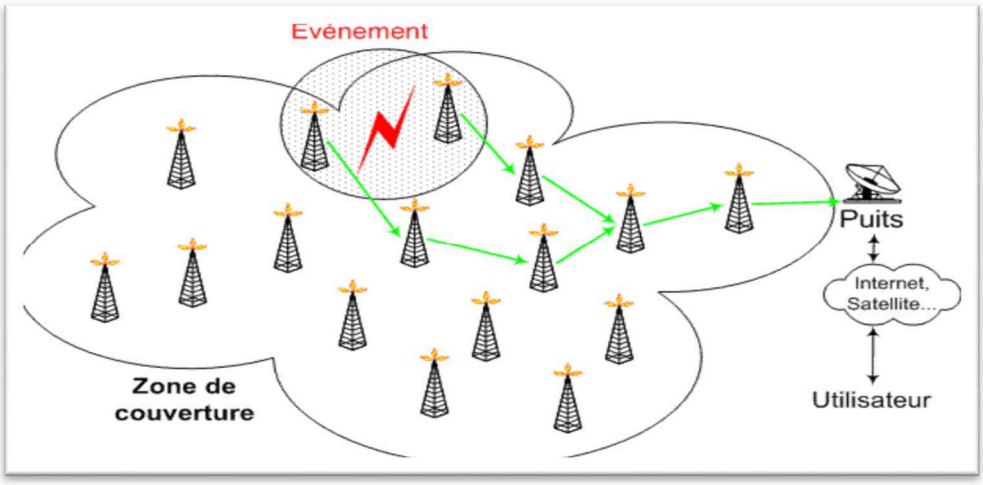

Figure I-5: Collecte d'informations suite à un événement

I.3.3 Contraintes de conception des RCSF

Les principaux facteurs et contraintes influençant l'architecture des réseaux de capteurs peuvent être résumés comme suit :

- **Tolérance aux pannes :** Les algorithmes et protocoles doivent tenir compte du fait qu'un nœud peut cesser de fonctionner par manque d'énergie ou parce qu'il a été détruit accidentellement par un animal ou intentionnellement par des personnes. Ils devront adapter leur niveau de tolérance aux pannes en fonction de l'hostilité du milieu dans lequel est déployé le réseau.
- **Le passage à l'échelle :** Le nombre de nœuds capteurs augmente sur un réseau sans fil et ce nombre peut atteindre des milliers voire plus d'un million. De ce fait, un nombre aussi important de nœuds engendre beaucoup de transmissions entre les nœuds et peut imposer des difficultés pour le transfert de données [4]. En outre, la plus des protocoles sont conçus pour des réseaux de taille moyenne et perdent leurs performances quand le nombre de nœuds augmente dans le réseau.
- **L'environnement :** Les capteurs sont souvent déployés en masse dans des endroits tels que des champs de bataille, à l'intérieur de grandes machines, au fond d'un océan, dans des champs biologiquement ou chimiquement souillés,... Par conséquent, ils doivent pouvoir fonctionner sans surveillance dans des régions géographiques éloignées.
- **La topologie de réseau :** Le déploiement d'un grand nombre de nœuds nécessite une maintenance de la topologie. Cette maintenance consiste en trois phases : déploiement, post-déploiement (les capteurs peuvent bouger, ne plus fonctionner,...) et redéploiement de nœuds additionnels quand le taux de couverture de la zone d'intérêt diminue au-dessous d'un certain seuil.
- **Les contraintes matérielles :** La principale contrainte matérielle est la taille du capteur. Les autres contraintes sont la consommation d'énergie qui doit être moindre pour que le réseau survive le plus longtemps possible, qu'il s'adapte aux différents environnements (fortes chaleurs, eau,..), qu'il soit autonome et très résistant vu qu'il est souvent déployé dans des environnements hostiles.
- **Le faible débit :** le débit garanti par les capteurs est trop limité. D'où, il faudrait optimiser l'information envoyé au centre de contrôle par exemple utiliser des nœuds agrégateurs pour agréger l'information localement avant de l'envoyer.
- **La consommation d'énergie :** Un capteur, de par sa taille, est limité en énergie (<1.2V). Dans la plupart des cas le remplacement de la batterie est impossible. Ce qui veut dire que la durée de vie d'un capteur dépend grandement de la durée de vie de la

batterie. Dans un réseau de capteurs (multi-sauts) chaque nœud collecte des données et les envoie. Le dysfonctionnement de quelques nœuds nécessite un changement de la topologie du réseau et un re-routage des paquets. Toutes ces opérations sont gourmandes en énergie, c'est pour cette raison que les recherches actuelles se concentrent principalement sur les moyens de réduire cette consommation [4].

I.3.4 Applications des RCSF

La diminution de taille et de coût des micro-capteurs, l'élargissement de la gamme des types de capteurs disponibles (thermique, optique, vibrations...) et l'évolution des supports de communication sans fil, ont élargi le champ d'applications des RCSF. Parmi ces applications, nous citons :

- **Domaine environnemental :** On peut créer un réseau autonome en dispersant les nœuds dans la nature. Des capteurs peuvent ainsi signaler des événements tels que feux de forêts, tempêtes ou inondations. Ceci permet une intervention beaucoup plus rapide et efficace des secours.
- **Détection d'intrusions:** En plaçant, à différents points stratégiques, des capteurs, on peut ainsi prévenir des cambriolages ou des passages de gibier sur une voie de chemin de fer (par exemple) sans avoir à recourir à de coûteux dispositifs de surveillance vidéo.
- **Contrôle de la pollution :** on pourrait disperser des capteurs au-dessus d'un emplacement industriel pour détecter et contrôler des fuites de gaz ou de produits chimiques. Ces applications permettraient de donner l'alerte en un temps record et de pouvoir suivre l'occurrence de l'événement.
- **Domaine d'agriculture :** des nœuds peuvent être semés avec des graines dans la terre. On peut ensuite consulter le réseau de capteurs sur l'état du champ (déterminer par exemple les secteurs les plus secs afin de les arroser en priorité) et par conséquent l'arrosage sera efficace et la quantité d'eau sera économisée en particulier dans les pays qui connaissent la sécheresse. On peut aussi imaginer équiper des troupeaux de bétail de capteurs pour connaître en tout temps, leur position ce qui éviterait aux éleveurs d'avoir recours à des chiens de garde.
- **Domaine médical :** 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.

- **Domaine militaire :** les réseaux de capteurs sont utilisés pour la surveillance, la détection des intrusions, la détection des substances dangereuses, etc. Ils peuvent être rapidement déployés et utilisés pour la surveillance des champs de bataille afin de fournir des renseignements concernant l'emplacement, le nombre, le mouvement, et l'identité des soldats et des véhicules, ou bien encore pour la détection des agents chimiques, biologiques et nucléaires.

I.4 Conclusion

Dans ce chapitre, nous avons présenté les réseaux de capteurs sans fil, leurs caractéristiques et les contraintes sous-jacentes au développement d'applications pour les RCSF. En outre, nous avons mis le point sur quelques facteurs permettant de concevoir des protocoles de routage tolérants aux pannes pour les RCSF.

Dans le chapitre qui suit, nous présentons les protocoles de routage tolérants aux pannes les plus utilisés dans les RCSF.

Chapitre II

La tolérance aux pannes dans les RCSF

Chapitre II

La tolérance aux pannes dans les RCSF

II.1 Introduction

Dans les réseaux de capteurs, on doit assurer la fidélité de détection c'est-à-dire tout point de la zone d'intérêt est couvert par au moins un capteur. En outre, on doit aussi garantir la fidélité de routage c'est-à-dire qu'il doit exister au moins un chemin entre tout capteur et la station de base. Cette caractéristique de routage nous a permis de concevoir plusieurs protocoles de routage tolérants aux pannes.

On distingue de types d'anomalies de fonctionnement dans les RCSF. La première quand il existe des capteurs qui remontent des données erronées à la station de base et la deuxième quand la donnée envoyée par un tel capteur n'arrive pas à la station de base.

Dans ce chapitre, nous présentons tout d'abord une taxinomie sur la tolérance aux pannes dans les RCSF. Puis, nous exposons les protocoles de routage tolérants aux pannes les plus utilisés pour les RCSF.

II.2 Taxonomie de la tolérance aux pannes

La limitation d'énergie dans les capteurs sans fil, les environnements hostiles comme le champ de bataille, dans lesquels ils pourraient être déployés, la perte de connexions sans fil 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, sont des facteurs qui rendent ce type de réseaux très sensibles aux pannes. De ce fait, il est donc nécessaire de considérer la tolérance aux pannes comme critères indispensable dans la conception des protocoles de routage pour les RCSF.

Dans cette section, nous présentons quelques notions sur la tolérance aux pannes et en particulier dans les RCSF.

II.2.1 Notion de panne

Une panne du système se produit lorsque son comportement devient inconsistant et ne fournit pas le résultat voulu. La panne est une conséquence d'une ou plusieurs erreurs

où une erreur représente un état invalide du système dû à une faute (défaut). La faute est donc la première cause de l'erreur, cette dernière provoque la faille du système [6].

II.2.2 Notion de tolérance aux pannes

Dans les RCSF, un ou plusieurs capteurs peuvent ne pas fonctionner correctement, la propriété de tolérance aux pannes est définie par l'aptitude du réseau à maintenir ses fonctionnalités, en cas de panne de certains de ses nœuds. Elle vise donc à minimiser l'influence de ces pannes sur la tâche globale du réseau.

II.2.3 Procédure générale de tolérance aux pannes

La conception d'une procédure pour la tolérance aux pannes dépend de l'architecture et des fonctionnalités du système. Cependant, certaines étapes générales sont exécutées dans la plupart des systèmes [7].

a) Détection d'erreurs

C'est la première phase dans chaque schéma de tolérance aux pannes, 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 alors que la détection en ligne vise l'identification de pannes en temps réel et elle est effectuée simultanément avec l'activité du système.

b) Détention de la panne

Cette phase établit des limites des effets de la panne sur une zone particulière afin d'empêcher la contamination des autres régions. En cas de détection d'intrusion, par exemple, l'isolation des composants compromis minimise le risque d'attaque des composants encore fonctionnels.

c) Recouvrement d'erreur

C'est la phase dans laquelle on effectue des opérations d'élimination des effets de pannes. Les deux techniques les plus utilisées sont "masquage de panne" qui utilise l'information redondante correcte pour éliminer l'impact de l'information erronée, et "répétition" qui effectue, après la détection d'une panne, un nouvel essai pour exécuter une partie du programme, dans l'espoir que la panne soit transitoire.

d) Traitement de pannes

Dans cette phase, la réparation du composant en panne isolé est effectuée. La procédure de réparation dépend du type de la panne. Les pannes permanentes exigent une substitution du composant avec un autre composant fonctionnel. Le système doit contenir un ensemble d'éléments redondants (ou en état standby) qui servent à remplacer les nœuds en panne.

II.3 Classification des protocoles de tolérance aux pannes

Les protocoles tolérants aux pannes peuvent être vus de plusieurs angles différents. De ce fait, un ensemble de critères est défini pour les classifier. Nous citons, entre autre, deux principales catégories : la classification temporelle et la classification architecturale.

II.3.1 Classification temporelle

Dans cette classification, nous divisons l'ensemble des algorithmes en deux catégories, et cela selon la phase de traitement. Si le traitement est effectué avant la panne on parle donc d'algorithmes préventifs sinon, les algorithmes sont dits curatifs.

a) Algorithme préventif

Ce type d'algorithmes implémente des techniques tolérantes aux pannes 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 nœuds.

b) Algorithme curatif

Un algorithme curatif utilise une approche optimiste, où le mécanisme de tolérance aux pannes implémenté n'est exécuté qu'après la détection de pannes. Pour cela, plusieurs algorithmes de recouvrement après pannes sont proposés dans la littérature, par exemple: le recouvrement du chemin de routage, l'élection d'un nouvel agrégateur dans une architecture clustérisée, etc.

II.3.2 Classification architecturale

Cette classification traite les différents types de gestion des composants, soit au niveau du capteur individuellement ou bien sur tout le réseau. Nous distinguons trois principales catégories :

a) Gestion de la batterie

Cette catégorie est considérée comme une approche préventive, où les protocoles définissent une distribution uniforme pour la dissipation d'énergie entre les différents nœuds capteurs ; afin de mieux gérer la consommation d'énergie et augmenter ainsi la durée de vie de tout le réseau. En outre, le mécanisme de mise en veille est une technique efficace de gestion de batterie. En effet, les protocoles déterminent des délais de mise en veille des nœuds capteurs inactifs pour une meilleure conservation d'énergie.

b) Gestion de flux

Cette catégorie regroupe les techniques qui définissent des protocoles de gestion de transfert des données (routage, sélection de canal de transmission, etc.). Nous pouvons trouver des approches préventives ou curatives sur les différentes couches (réseau, liaison de données, etc.) telles que :

- **Routage multipath**: utilise un algorithme préventif pour déterminer plusieurs chemins depuis chaque capteur vers le nœud collecteur c'est-à-dire le réseau devrait être de type k-connected. Ceci garantit la présence de plus d'un chemin fiable pour la transmission et offre une reprise rapide du transfert en cas de panne sur le chemin principal et choisissant un des chemins qui restent.
- **Recouvrement de route** : après détection de panne, une technique curative permet de créer un nouveau chemin qui soit le plus fiable pour retransmettre les données.
- **Allocation de canal** : cette solution est implémentée au niveau de la couche MAC. Elle permet d'effectuer une allocation du canal de transmission d'une manière à diminuer les interférences entre les nœuds voisins et éviter les collisions durant le transfert.
- **Mobilité** : certains protocoles proposent comme solution tolérante aux pannes la sélection d'un ensemble de nœuds mobiles chargés de se déplacer entre les capteurs et 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 simple.

c) Gestion des données

Les protocoles classés dans cette catégorie offrent une meilleure gestion de données et de leur traitement. Deux principales sous-catégories sont déterminées :

- **Agrégation** : considérée comme approche préventive, l'opération d'agrégation effectue un traitement supplémentaire sur les données brutes captées depuis l'environnement. Un nœud agrégateur combine les données provenant de plusieurs nœuds en une information significative. Ce qui réduit considérablement la quantité de données transmises en consommant moins d'énergie pour leur dissémination. Ceci permet donc d'augmenter la durée de vie du réseau.
- **Clustering** : une des importantes approches pour traiter la structure d'un réseau de capteurs est le clustering. Il permet la formation d'un backbone virtuel qui améliore l'utilisation des ressources rares telles que la bande passante et l'énergie. Par ailleurs, le clustering aide à réaliser du multiplexage entre différents clusters. En outre, il améliore les performances des algorithmes de routage. Plusieurs protocoles utilisent cette approche préventive et parfois elle est considérée comme une approche curative.

II.4 Approches de tolérance aux pannes dans les RCSF

II.4.1 Approche de Checkpoint

L'approche de checkpoint est une technique qui permet de garantir la tolérance aux pannes dans les systèmes non fiables et les systèmes distribués [8]. Elle permet de sauvegarder l'état de tout le système et le récupère dans le cas de l'occurrence d'une anomalie. Cependant, elle requiert des supports de stockage consistants en termes de mémoire. C'est la raison pour laquelle il est un peu difficile à l'appliquer dans le domaine des RCSF.

II.4.2 Solutions de routage pour la tolérance aux pannes

Dans les RCSF, les capteurs sont sujets à des pannes à cause de l'épuisement de leurs batteries, ou l'écrasement par des animaux, etc. Il résulte de l'occurrence des pannes une difficulté pour acheminer les données collectées à la station de base. Pour remédier à cette problématique des protocoles de routage tolérants aux pannes ont été proposés dans la littérature.

La tolérance aux pannes pour assurer une fiabilité de délivrance de paquets à la station de base est traitée au niveau de la couche réseau. Dans ce qui suit, nous présentons les fonctionnalités de certains protocoles de routage tolérants aux pannes et nous discutons leurs limites :

a) Protocole de routage dynamique tolérant aux pannes

L'objectif de ce protocole est de maintenir la connectivité du réseau, même si un nœud est sur le point d'épuiser son énergie pour assurer la livraison de données à la station de base tout en prolongeant la durée de vie du réseau [9].

Dans ce protocole, quand un nœud capteur est sur le point d'épuiser son énergie, il essaie de trouver un chemin alternatif pour établir une nouvelle connexion avec ses nœuds voisins. Ce chemin alternatif augmente la fiabilité de transmission de données entre les nœuds source et leurs voisins dans la direction de la station de base qui relaient les paquets envoyés par ces derniers. Ce protocole s'exécute en trois phases :

- **Mise en œuvre et établissement de chemin :** Chaque nœud est caractérisé par son identifiant nœud (N_j), le niveau (HC_j), nœud parent (P_j), un tableau (A_j) pour stocker les paquets de données jusqu'à ce qu'un accusé de réception soit reçu. La station de base est initialisé avec HC = 0, P = BS, tandis que les nœuds ordinaires avec d'autres $HC_j = \infty$, $P_j = 1$.

  Une fois les nœuds sont déployés, la station de base diffuse un message d'avertissement ADVT (N_j, HC_j) pour découvrir les nœuds qui sont voisins à la station de base. Ces nœuds sont considérés comme des nœuds de niveau 1 puisqu'ils se trouvent à un saut de la station de base qui est considérée comme un nœud parent pour ces nœuds de niveau 1. Lorsqu'un nœud reçoit un message ADVT, son HC sera augmenté de un que de celui qui lui a envoyé le message ADVT et il est considéré comme un nœud de niveau N+1 si le nombre de sauts reçus est N. Ainsi, le message ADVT est utilisé pour hiérarchiser le réseau en des niveaux relativement à la station de base.

- **Transmission de données :** Une fois que la hiérarchisation de niveaux est établie, la phase de transmission de données commence. De ce fait, lorsqu'un événement survient au niveau du nœud source. Ce dernier transmet le paquet de données relatif à l'événement au nœud parent et stocke une copie de ce paquet de données. Quand un parent reçoit le paquet de données émis, il envoie un accusé de réception (ACK) au

nœud qui a transmis le paquet. De son côté le nœud source, une fois qu'il reçoit le paquet ACK, il supprime la copie du paquet de données correspondant. Cela continue jusqu'à ce que la station de base reçoive le paquet de données. Un numéro de séquence est attribué à chaque paquet de données transmis pour assurer la fiabilité et garantir sa livraison à la station de base. Si un paquet de données est perdu, il pourra être récupéré à partir du dernier nœud expéditeur.

- **Rétablissement de chemin :** Si un nœud est sur le point d'épuiser son énergie, il envoie un message de notification à ses voisins fils en leur demandant de changer leurs nœuds parents pour maintenir la connectivité. Les nœuds fils qui reçoivent ce message, utilisent des paquets « Hello » pour découvrir les nouveaux parents dans leurs voisinages. Les nœuds fils modifient leurs paramètres N_j, HC_j en fonction de la réponse au message Hello. Si la réponse provient d'un nœud de niveau inférieur, les nœuds fils gardent leur N_j sinon c'est-à-dire le message provient d'un nœud voisin, les nœuds fils doivent incrémenter leurs niveaux de 1.

La limitation de ce protocole est que le temps pris pour trouver un nouveau nœud voisin affecte la durée de livraison de données.

b) Protocole de routage tolérant aux pannes multi-niveaux (FMS)

Le protocole FMS [10] permet de maintenir la connectivité du réseau, même si un nœud est sur le point d'épuiser son énergie. Il permet aussi d'assurer la fiabilité et la rapidité de livraison des données à la station de base car il est conçu pour les applications orientées événement. Généralement, les capteurs sont déployés aléatoirement et en grand nombre. De ce fait, il y aura une redondance dans la livraison de données ce qui a une conséquence sur la durée de vie du réseau de capteurs. Pour remédier à cette limite, FMS permet un ordonnancement d'activité des capteurs en passant un certain nombre de capteurs en mode « veille » sans affecter la fiabilité de livraison de données. Ceci est dans le but d'économiser l'énergie.

Dans FMS, on suppose que chaque nœud possède un identifiant unique, dénoté (Nr), et la communication entre les nœuds voisins est bidirectionnelle. En outre, on suppose que les nœuds sont contraints en termes de puissance de traitement, de stockage et de l'énergie, tandis que la station de base est considérée comme un nœud qui a plus de ressources pour effectuer des tâches ou de communiquer avec les autres nœuds.

Le protocole FMS effectue deux opérations de base :

- **Détermination des niveaux des nœuds et établissement de chemin :** cette phase est analogue à celle du protocole cité ci-dessus.
- **Ordonnancement d'activité des capteurs et transmission de données :** l'ordonnancement d'activité des capteurs consiste à faire passer un certain nombre de nœuds périodiquement en mode veille. Au cours de cette période, les nœuds actifs transmettent les paquets de données. Avant qu'un nœud passe en mode veille, il devra informer ses nœuds fils afin qu'ils choisissent un autre nœud parent pour relayer les données. En outre, quand un nœud est en mode veille, il passera en mode actif que si son énergie est supérieure à une certaine valeur seuil. Le choix des nœuds actifs se fait aléatoirement et d'une manière périodique pour que le nœud n'épuise pas son énergie rapidement. Quand un nœud est en mode actif, il est participe à l'opération de transmission de données à la station de base. De ce fait, la connectivité est toujours maintenue même si un nœud est mis en mode veille ou il est sur le point de perde son énergie. Ainsi, FMS est considéré comme un protocole fiable et tolérant aux pannes.

FMS présente les mêmes limitations que le protocole cité précédemment. Il est performant dans un environnement optimal mais ses performances se dégradent dans un environnement réel.

c) Protocole de routage adaptatif tolérant aux pannes (RERP)

Dans RERP [11], il est supposé que chaque nœud a au moins deux voisins dans la direction vers la station de base. Par conséquent, il aura au moins deux chemins alternatifs pour acheminer les données vers la station de base. Ainsi, la capacité d'un nœud tolérant aux pannes dépend du nombre des nœuds voisins actifs c'est-à-dire si un nœud a N voisin, il peut tolérer N-1 nœuds en panne.

Le protocole RERP comporte deux tâches :

- **Mise en place de RERP** : cette tâche s'exécute en cinq phases :
  - **Phase de publicité :** Dans cette phase, la station de base diffuse un paquet de publicité à ses nœuds voisins pour indiquer qu'elle peut recevoir des paquets de données. Les nœuds qui reçoivent le paquet de publicité établissent une table de routage pour indiquer le chemin vers la station de base.
  - **Phase d'initialisation :** Dans cette phase, les nœuds qui n'ont pas de chemin direct vers la station de base diffusent une requête de découverte de routes (RREQ) vers la station de base. Quand un concentrateur reçoit le paquet (RREQ), il diffuse une

réponse (RREP). De même si c'est ce nœud a déjà reçu la requête (RREQ) il diffuse un paquet (RREP) s'il existe un chemin entre lui et le concentrateur, sinon le paquet (RREQ) sera ignoré.

- **Route de sélection :** Une table de routage est utilisée pour construire et entretenir les routes. Le choix de l'itinéraire des nœuds relais est basé sur l'énergie restante des nœuds.
- **Phase de transfert de données :** Les nœuds capteurs génèrent des paquets de données à chaque fois qu'ils détectent toute nouvelle information. Cette information est transmise à la station de base en un mode multi-sauts.
- **Table de sauvegarde :** En plus du chemin principal, un chemin alternatif est prévu pour tous les nœuds du réseau. Chaque fois qu'un nœud reçoit un paquet RREP, s'il ne dispose pas de chemin direct vers la station de base, il stocke le chemin dans la table de routage, et il stocke les paquets (RREP) dans une table de sauvegarde. La table de routage de secours dispose de deux champs, l'identifiant du nœud ID et son énergie.

- **Rapport d'erreurs** : nous distinguons les messages d'erreurs suivants :
  - **Echec des liens :** le message d'échec de liens est généré dans les deux cas. Le premier se produit quand un RTS est envoyé mais aucun CTS correspondant n'est reçu et le nombre maximal de tentatives est dépassé. Le second se passe quand un paquet de données a été transmis, mais il n'a jamais reçu un ACK et le nombre maximal de tentatives est dépassé.
  - **Message de batterie critique :** Ce message est généré lorsque le niveau de la batterie d'un nœud est inférieur à une valeur seuil dite critique. Ce message est envoyé au nœud source qui a envoyé les données et également aux voisins de ce nœud. Quand les autres nœuds reçoivent ce message ils suppriment l'identifiant du nœud défaillant de leurs tables de routage ou de leur table de voisins.
  - **Message de destination inaccessible** : Ce message est généré lorsque le paquet de données est mis au rebut sans être transmis au nœud de destination en raison de l'indisponibilité du chemin vers la station de base.
  - **Sélection du chemin de secours :** Chaque nœud possède une table de routage de secours dans laquelle il stocke un chemin de secours vers la destination. Quand un

nœud échoue dans la transmission de paquet de donnée, alors son voisin consulte la table de secours pour trouver le chemin alternatif afin qu'il puisse transmettre le paquet de données.

Dans RERP la communication entre les nœuds est réalisée par des messages Requête/Réponse. Ce type de messages est utilisé pour vérifier si le voisin est accessible et pour calculer le temps de parcours.

RERP présente certaines limitations telles que la consommation d'énergie qui est assez grande lors de la diffusion des rapports d'erreurs.

d) Protocole de routage temps réel tolérant aux pannes (DMRF)

DMRF [12] fonctionne en deux modes de transmission de données : saut-à -saut et "Jumping". Chaque nœud utilise le temps restant pour transmettre un paquet à la station de base et l'ensemble des nœuds de transfert FCS (Set candidat Forwarding) pour choisir dynamiquement le prochain saut. Quand un nœud présente une défaillance, alors la congestion du réseau ou une région vide se produit. Le mode de transmission sera passé en mode "Jumping", ce qui peut réduire le délai de transmission, et assure la fiabilité de la livraison des paquets de données envoyés à la station de base dans un délai spécifié.

Dans DMPF, le processus de transmission est divisé en cinq étapes :

- **Phase d'initialisation :** Dans cette phase, DMRF initialise la liste de voisinage des nœuds, la liste de l'état du réseau (information sur la congestion d'un nœud, les zones vides, …), la liste des candidats FCS, la table des probabilités de transition, et la voie de transmission initiale.
- **Phase de transmission des données :** Dans cette phase, DMRF détecte la défaillance d'un nœud, la congestion du réseau ou une région vide. Le temps restant pour acheminer un paquet de données jusqu'à la station de base sera contrôlé. A partir de ce temps, le paquet sera transmis en mode "Jumping" ou non. Si aucune des conditions ci-dessus ne s'est produite, DMRF sélectionne dynamiquement un membre du FCS comme nœud relais. En outre, une fois les nœuds défaillants sont détectés, ou le temps restant est inférieur à un certain seuil, le mode de transmission "Jumping" sera utilisé.
- **Phase de transmission "Jumping" :** au cours de cette phase, chaque nœud ajuste dynamiquement le contenu de FCS et calcule la probabilité pour transiter par chacun

de ces nœuds. Dans ce mode, le paquet de données peut utiliser un saut d'une grande portée pour éviter les nœuds défaillants. Cependant, il ne peut pas garantir le succès de la transmission. Donc, la phase d'ajustement des probabilités de transitions est effectué après chaque transmission "Jumping".

- **La phase d'ajustement des probabilités :** Dans cette phase, DMRF ajuste la probabilité de saut en fonction du résultat de la transmission "Jumping" (succès ou l'échec) et renvoie l'information à son nœud en amont. Lorsque le paquet de données arrive au nœud récepteur, on considère que la transmission est terminée.

DMRF présente certaines limitations en particulier dans le mode "Jumping" qui ne garantit la fiabilité de livraison de données et qui consomme plus d'énergie quand il utilise une grande portée.

e) AODV tolérant aux pannes (ENFAT-AODV)

ENFAT-AODV [13] est un protocole de routage qui permet la tolérance aux pannes, l'auto-démarrage et le routage multi-sauts entre les nœuds. ENFAT-AODV établit les plus courts chemins entre les nœuds en nombre de sauts. En outre, il permet aussi aux nœuds d'établir un chemin de secours quand le chemin principal présente des ruptures. Toutefois, dans ENFAT-AODV, il y a des champs additionnels comparativement à la version originale du "AODV". Certains champs sont ajoutés dans les paquets de contrôle tels que "BACKUP" (dans RREQ et RREP), "UPDATE" dans RREQ et "Distance" dans RREQ.

ENFAT-AODV réduit également la complexité de mise en œuvre par suppression des paquets de contrôle inutiles dans le réseau et il s'exécute comme suit :

- **Découverte du chemin principal :** Quand un chemin principal de livraison des données vers la station de base est nécessaire, le nœud source diffuse un message de découverte de chemin principal vers la station de base en utilisant des messages RREQ. Ainsi, chaque nœud intermédiaire recevant le message RREQ établit un chemin inverse vers le nœud source. Si le nœud reçoit le message RREQ pour la première fois et si ce dernier ne connaît pas la route principale menant à la destination, il transmettra le message RREQ à ses voisins. Si le nœud de réception est la destination ou bien s'il connaît la route principale menant à la destination, il va générer un itinéraire principal (RREP principal). Ensuite, ce RREP principal

est envoyé à en un saut à la source. Lorsque la source reçoit le message RREP, cette dernière enregistre la route principale menant à la destination dans sa table de routage.

- Construction du chemin de secours : Au cours de la phase de découverte de chemin principal, les nœuds d'un chemin principal qui reçoivent un RREP principal créent un chemin de secours vers la station de base en diffusant un paquet de secours RREQ. Après la diffusion du RREQ de secours, le nœud attendra un paquet RREP de secours de la part de la destination ou d'un nœud intermédiaire qui peut satisfaire les conditions suivantes :

  - Il dispose d'une entrée de secours dans le chemin principal vers la station de base,
  - Il n'est pas un nœud sur le chemin principal,
  - Le nombre de sauts du chemin de secours à partir du nœud intermédiaire à la destination est inférieur aux autres nœuds.

- **Entretien de la route :** Pendant la période de livraison des paquets de données quand le chemin principal n'est pas valide, le nœud utilise immédiatement sa route de secours pour relayer les prochains paquets de données. Par la suite, le nœud sur le nouveau chemin principal, qui utilise une route de secours, dirige un processus "Découverte de route de secours" visant à trouver un autre chemin. Par conséquent, il augmente la fiabilité et la disponibilité par rapport à la première version de "AODV".

La limitation de ce protocole concerne essentiellement la consommation de l'énergie à cause de l'inondation des messages de contrôle.

f) Diffusion dirigée pour la tolérance aux pannes (FaT2D)

FaT2D (Fault Tolerant Directed Diffusion for Wireless Sensor Networks) [14] est un protocole de tolérance aux pannes basée sur la diffusion et il fournit une tolérance aux pannes grâce à la construction de plusieurs chemins entre les nœuds et l'exploration périodique des routes pour la découverte d'éventuelles anomalies au niveau de ces routes. FaT2D définit une nouvelle technique qui permet de détecter rapidement une panne et la recouvrir quand il y a une collision entre les nœuds et des changements de topologie. Il s'exécute selon la démarche qui suit :

- **Détection de panne** : FaT2D introduit un nouveau délai d'attente de détection de pannes, noté TFD, est défini afin de réduire le temps de recouvrement de la panne et par conséquent le remplacement des nœuds défaillants.

- Si TFD s'épuise, FaT2D transmet immédiatement un nouveau message appelé ”Explore-Request“ pour notifier l'événement de détection de la panne et demande une nouvelle exploration pour trouver un autre chemin fiable qui remplace le chemin défaillant. Par conséquent, tout nœud appartenant au chemin défaillant supprime le gradient correspondant pour éliminer la panne.
- **Recouvrement du chemin :** Quand TFD est épuisé, il déclare la défaillance d'un nœud. De ce fait, FaT2D lance un processus pour réparer le chemin défectueux en envoyant un message de demande d'exploration appelé “ExploreRequest”. Ce message contient les informations sur la route défectueuse et il est acheminé pour atteindre le nœud cible sans utiliser les transmissions bouclées ou rechercher les nœuds non adéquats. Quand le nœud cible reçoit le message “ExploreRequest”, il ne le transmet pas, puis il lance une exploration par inondation comme dans “Direct Diffusion”. Cela génère une phase d’exploration afin de trouver un nouveau chemin fiable.
- **Elimination des pannes :** Pour chaque nœud intermédiaire recevant le message “ExploreRequest”, FaT2D vérifie si ce nœud appartient au chemin défectueux. Si c'est le cas, il aura un effet négatif pour renforcer son gradient. Ce dernier sera réélu par une exploration lancée par le nœud source du chemin correspondant. Ainsi, chaque nœud demande à ces voisins en amont de supprimer le chemin brisé et arrêter l'envoi de données sur ce chemin.

g) Algorithme PEQ (Periodic, Event-driven, Query-based)

PEQ [15] combine la conservation d'énergie avec le routage multi-chemins en sélectionnant parmi tous les chemins disponibles, ceux qui consomment moins d'énergie et le considère comme étant chemin principal et les autres des chemins secondaires. En plus de ce mécanisme préventif qui permet un routage fiable avant l’occurrence des pannes, un mécanisme de recouvrement de pannes est implémenté. Ce dernier remplace le chemin défaillant par un autre chemin qui a des liens fiables et consomme moins d'énergie.

PEQ introduit le paradigme Publish/Subscribe comme montre la figure II-1 pour l'interaction entre le collecteur et les capteurs simples. En effet, les capteurs envoient des notifications d'événements au collecteur, qui va souscrire son intérêt pour certaines de ces informations. Les capteurs concernés publient par la suite l'information désirée.

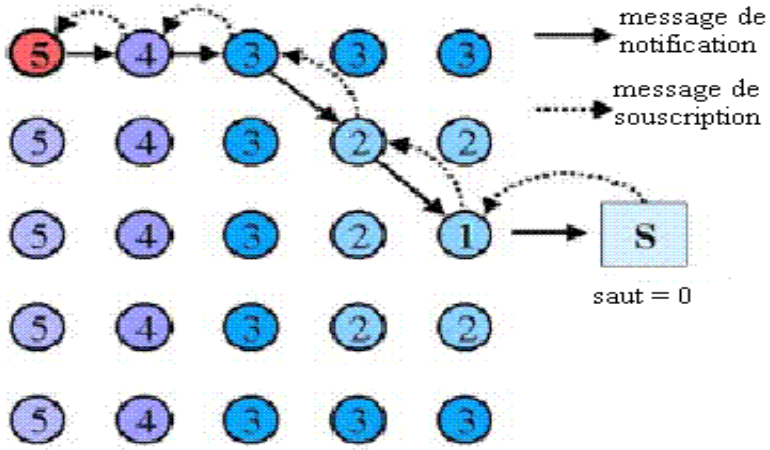

Figure II-1: Mécanisme Publication/Souscription [15]

PEQ s'exécute comme suit :

- **Construction de l'arbre de routage :** cet arbre permet de définir les différents chemins multi-sauts possibles pour acheminer les données de chaque nœud à la station de base. Le collecteur commence le processus en initialisant la variable « saut » à 0; par la suite, chaque capteur prend la valeur du saut actuelle, l'incrémente puis l'envoie à tous ses voisins. Ainsi la valeur au niveau de chaque capteur désigne le nombre nécessaire de sauts pour communiquer avec le collecteur. A la fin de cette phase seulement les meilleurs chemins sont enregistrés.
- **Transmission de paquets de notification** : chaque capteur envoie selon sa table de routage et l'événement capté, une notification de l'information qu'il a à sa disposition. Pour cela, il utilise le chemin le plus court et le moins coûteux en terme d'énergie.
- **Propagation des paquets de souscription** : dans cette étape, après une souscription, par le collecteur, des données à transmettre, chaque capteur achemine cette dernière jusqu'au capteur concerné.
- **Mécanisme de recouvrement de route :** le recouvrement est effectué après détection de pannes (figure II-2). Un capteur envoie son paquet puis attend un acquittement ACK. S'il le reçoit, le message a été bien transmis ; sinon une panne est détectée au niveau du chemin de routage. On effectue donc une recherche "SEARCH" pour la sélection d'un autre capteur destination tout en minimisant le coût du nouveau chemin. Si aucun capteur n'est trouvé le capteur devient isolé et doit donc augmenter son rayon de transmission radio pour atteindre les capteurs voisins lointains.

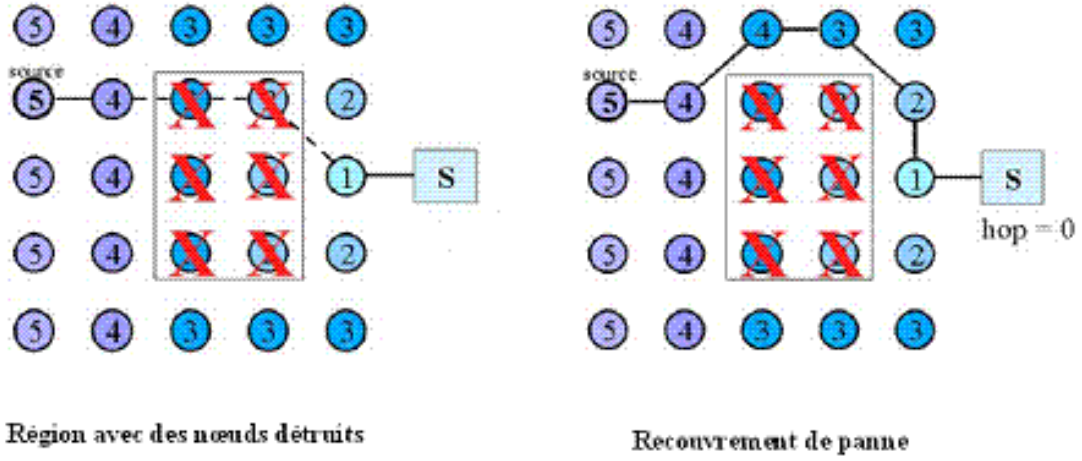

Figure II-2: Recouvrement de routes dans PEQ [15]

h) Protocole EAR

EAR (An **E**nergy and **A**ctivity Aware **R**outing Protocol for Wireless Sensor Networks in Smart Environments) [16] propose une solution hybride pour la tolérance aux pannes. Il permet une meilleure conservation d'énergie et définit plusieurs chemins de routage afin de garantir une fiabilité de livraison de données. En outre, un mécanisme de recouvrement de pannes est implémenté.

Le protocole EAR supporte des réseaux de capteurs à collecteurs multiples. Chaque capteur génère un paquet RPT contenant des informations pour les préférences de l'utilisateur. Les paquets RPT peuvent être envoyés vers n'importe quel collecteur. Cependant, pour chaque capteur intermédiaire le protocole de routage choisit le meilleur chemin qui réduit la consommation d'énergie et la latence. EAR s'exécute selon les étapes suivantes :

- **Phase d'initialisation :** Cette phase permet la construction de l'arbre de routage contenant tous les chemins possibles pour la dissémination des données. Chaque collecteur diffuse un message d'avertissement ADV demandant des paquets RPT. Seuls les capteurs voisins du collecteur qui reçoivent le message ADV, enregistrent le chemin dans leur table de routage ; sans qu'ils propagent le message ADV vers les autres capteurs, comme le montrent les étapes a) et b) de la figure II-3. Les autres capteurs envoient une demande RREQ (Route Request) cherchant un chemin vers le collecteur (étape c). Si un capteur ayant déjà une route stockée dans sa table, reçoit

RREQ, il envoie un paquet RREP (Route Reply) à son capteur voisin concerné par la demande (étapes d, e). Le processus d'initialisation se termine quand chaque capteur reçoit une réponse RREP suite à sa requête RREQ ; puis enregistre le chemin dans sa table de routage.

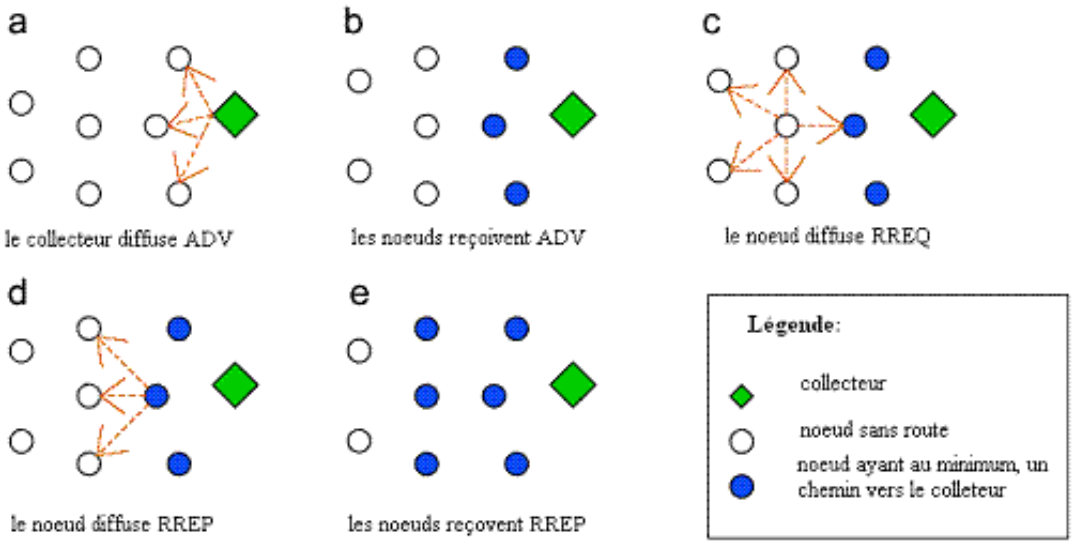

Figure II-3: Fonctionnement du protocole EAR [16]

- **Phase de gestion de route :** Les micro-capteurs, avec leur mémoire de taille réduite, ne peuvent pas garder tous les chemins possibles dans leurs tables de routage. Pour cela, et afin d'assurer une bonne tolérance aux pannes, on devrait garder que les meilleurs chemins. Le protocole EAR définit donc deux métriques pour la sélection des meilleurs chemins à mémoriser. La première métrique est le nombre de sauts dans une route. Ceci permet de choisir le chemin le plus court. Cependant, la qualité des liens n'est pas prise en considération ; dans ce cas, le plus court chemin n'assure pas forcément la fiabilité de transmission. En effet, si un chemin échoue à transmettre N paquets consécutifs, il sera mis dans une “liste noire” l'écartant ainsi d'une future utilisation. La deuxième métrique, appelée Score de route, est définie comme suit :

$$RS = PE \times WE + PT \times WT$$

  - PE: niveau de l'énergie du capteur du prochain saut ;
  - WE: poids assigné à PE dans l’intervalle [0-1] ;
  - PT: taux de succès dans la transmission ;
  - WT: poids assigné à PT dans [0-1] tel que (WT + WE = 1);
- **Phase dissémination de données :** Après les deux premières étapes, chaque capteur aura au moins un chemin vers le collecteur. Les capteurs commencent donc à générer

des paquets RPT, et le routage des données utilise la métrique “score de route” pour définir le meilleur chemin à emprunter. En cas où ce dernier présente une panne au niveau d'un ou plusieurs de ses capteurs, un mécanisme de recouvrement de route est exécuté, afin d'élire un second chemin fiable pour transmettre les données depuis le capteur vers le collecteur. Par ailleurs, au moment de sa durée d'inactivité, chaque capteur est mis en veille afin d'épargner davantage son énergie et augmenter ainsi la durée de vie de tout le réseau.

i) VTRP (Variable Transmission Range Protocol)

VTRP [17] est une solution d’ajustement du rayon de transmission pour une meilleure propagation de données. Il permet de remédier au problème d'obstacles en les évitant par l'augmentation du rayon de transmission. Ce dernier augmente la probabilité d'atteindre des capteurs actifs quand le rayon actuel utilisé ne couvre aucun capteur à cause de pannes ou d'inactivité des capteurs voisins ou encore dans le cas des réseaux à faible densité. En outre, VTRP offre une meilleure longévité du réseau en évitant l'utilisation fréquente des capteurs critiques (les voisins proches du collecteur) ceci permet d'alléger leur fonction de routage ; conserve leurs batteries et augmente ainsi la durée de vie de tout le réseau. VTRP s’exécute comme suit :

- **Phase de recherche :** Soient p_1 et p_2 deux capteurs du réseau. Dans la phase de recherche, p_2 utilise une diffusion périodique de message afin de découvrir le nœud p_1 le plus proche du collecteur. Cependant, une détection de panne est possible si aucun nœud p_1 n'est trouvé. Cet échec est causé par l’une des raisons suivantes : soit le nœud p_1 est mis en veille ; soit il est en panne ou bien à cause d'un obstacle qui empêche la communication entre p_1 et p_2.
- **Phase de transmission directe :** En cas où la phase de recherche se termine avec succès, le capteur p_2 envoie l'information au capteur p_1.
- **Phase d’ajustement du rayon de transmission :** Si la phase de recherche échoue (aucun nœud p n'est détecté) p_2 passe à la phase d’ajustement de son rayon de transmission qui représente le cas de recouvrement après pannes. En effet, chaque capteur maintient un compteur local initialisé à 0. A chaque échec de l'étape de recherche, le compteur est incrémenté, et le rayon de transmission R est modifié. Quatre différentes fonctions sont définies selon la vitesse de variation du rayon de transmission : linéaire, multiplicative, exponentielle et aléatoire :

- **Progrès constant :** VTRP est convenable dans ce cas des réseaux où un grand nombre de capteurs est compromis ;
- **Progrès multiplicatif :** $VTRP_m$ définit un rayon de transmission qui est augmenté d'une manière radicale. Ce changement offre une meilleure probabilité pour trouver des capteurs actifs. En revanche, il requiert une consommation d'énergie plus importante.
- **Progrès exponentiel** *:* $VTRP_p$ est une variante qui augmente le rayon d'une vitesse encore plus rapide ;
- **Progrès aléatoire :** quand la densité du réseau n'est pas connue au préalable, on utilise l'approche aléatoire $VTRP_r$ pour éviter un mauvais comportement du réseau suite à un mauvais choix.

II.4.3 Solutions basées sur le clustering pour la tolérance aux pannes

Les algorithmes proposés dans cette catégorie permettent d'améliorer les performances du processus d'auto-organisation du réseau. Les protocoles du clustering divisent le réseau en un ensemble de clusters ayant chacun un clusterhead qui récupère les données depuis tous les capteurs de son cluster puis les achemine vers le collecteur. Cette solution permet d'alléger la quantité d'informations qui circule, en effectuant des traitements au sein du cluster avant de propager les données vers le reste du réseau pour les transmettre au collecteur.

a) Protocole CPEQ

En plus des mécanismes de tolérance aux pannes implémentés dans PEQ [18], la variante CPEQ (Cluster-based PEQ) [18] utilise l'approche de clustering pour offrir une meilleure gestion de routage. En effet, les capteurs ayant le plus d'énergie résiduelle sont sélectionnés comme des nœuds agrégateurs (clusterheads). Dans un cluster, les nœuds membres à ce dernier envoient leurs données au clusterhead qui effectue d'éventuel traitement sur les données avant de les acheminer vers la station de base. Chaque capteur du réseau peut devenir clusterhead pendant une certaine période de temps selon son niveau de batterie. Le but principal de CPEQ est de distribuer d'une manière uniforme la dissipation d'énergie entre les capteurs, et de réduire la latence et le trafic de données dans le réseau. Le protocole CPEQ s'exécute en cinq étapes :

- **Configuration initiale :** Cette phase est exécutée de la même manière que dans l'algorithme PEQ ; où chaque capteur commence par un mécanisme de diffusion pour connaître par la suite le nombre de sauts nécessaires pour atteindre le collecteur le plus proche. En outre, CPEQ introduit un champ additionnel contenant le pourcentage des capteurs qui deviendront agrégateurs.
- **Sélection d'agrégateur :** C'est la phase d'élection des clusterheads. Après la configuration initiale, chaque capteur peut devenir agrégateur avec un pourcentage donné. En effet, chaque capteur génère un nombre aléatoire entre 0 et 1. Si ce nombre est inférieur à une probabilité p (probabilité pour devenir agrégateur), le capteur demande à tous ses voisins directs leur niveau de batterie en envoyant un paquet REQ_EN (Request Energy). Chaque voisin répond par un message REP_EN (Reply Energy) contenant son ID et la quantité d'énergie. Le capteur choisit le voisin ayant le maximum d'énergie et diffuse un SET_AGR (Set Aggregator) pour informer tous les capteurs du nouvel agrégateur. Les trois étapes de cette phase sont illustrées dans la figure II-4.

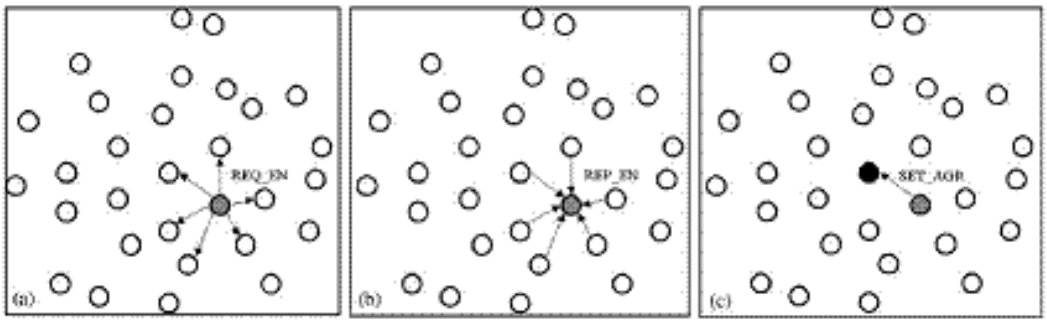

Figure II-4: Processus d'élection des clusterheads

- **Configuration de clusters :** Au cours de cette phase on assiste à la formation des clusters. Le nouveau capteur agrégateur sélectionné doit aviser ses voisins de son rôle d'agrégateur. De ce fait, chaque agrégateur construit son cluster. La configuration des clusters est réalisée à l'aide des messages AGR_NTF (Aggregator Notification) avec un champ TTL pour limiter la propagation du paquet sur les capteurs se trouvant à une distance inférieure ou égale au TTL. Chaque fois qu'un capteur reçoit ce message, il enregistre l'ID du capteur émetteur dans sa table de routage pour déterminer le chemin vers l'agrégateur. Si un capteur reçoit plusieurs messages AGR_NTF ; il choisit l'agrégateur avec le moindre nombre de sauts. La figure II-5 illustre la configuration de clusters avec un TTL=2.

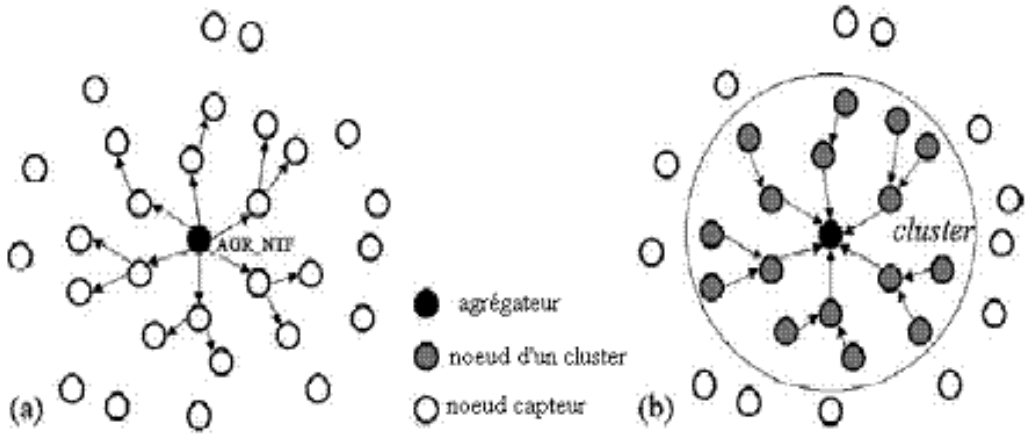

Figure II-5: Formation des clusters

- **Transmission de données à l'agrégateur :** Chaque capteur utilise sa table de routage pour envoyer la donnée vers son agrégateur. Dans CPEQ, l'agrégateur peut être considéré comme un nœud puits. Le mécanisme de recouvrement de chemin est aussi hérité du protocole PEQ.

- **Transmission de données au collecteur** : Après réception des données depuis les capteurs de son cluster, l'agrégateur doit acheminer ces données au collecteur. CPEQ utilise une communication multi-sauts entre l'agrégateur et le collecteur comme montre la figure II-6.

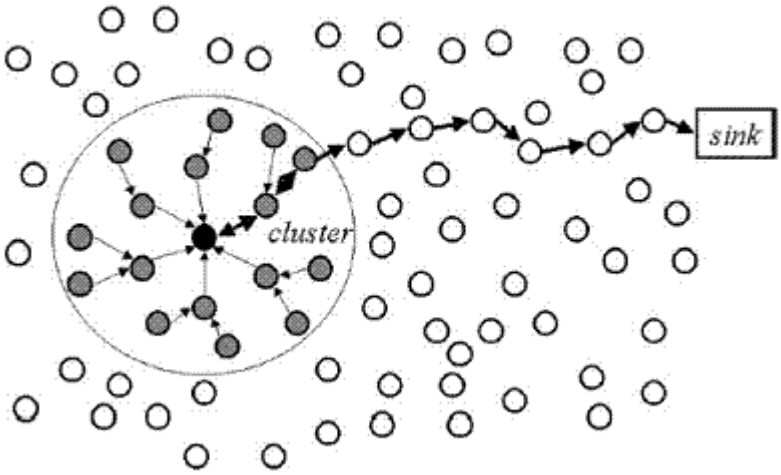

Figure II-6: Transmission des données au collecteur

b) Algorithme K-CDS [19]

On modélise un réseau de capteurs par le graphe G = (V, E) où V est l'ensemble des capteurs et E est l'ensemble des liens sans fil entre ces capteurs.

- **Définition 1 :** un réseau G est k-connexe si chaque deux capteurs du graphe sont connectés par au moins k chemins disjoints. Par conséquent, le réseau peut tolérer la défaillance d'un nombre de capteurs inférieur à k.
- **Définition 2 :** un sous-ensemble V' , V est un k-DS (k-dominating Set) de G si chaque nœud de V qui n'appartient pas à V' a au moins k voisins dans V'. Le k-DS V' devient k-CDS si le sous-graphe G = (V', E') est k-connexe.

L'algorithme K-CDS utilise une approche préventive basée sur le clustering. Il propose une construction d'un ensemble k-connexe dominant k-CDS comme un backbone virtuel pour offrir une efficacité de routage aussi bien qu'une bonne tolérance aux pannes. Pour cela, quatre approches ont été introduites ; dont deux sont des algorithmes probabilistes, une est déterministe et la dernière est une hybridation des approches déterministes et probabilistes.

c) KAT-Mobility

Dans KAT-mobiliy (K-means And TSP-based mobility) [20], en plus du concept de clustering, le concept de mobilité est implémenté au niveau des stations de base. Ces deux mécanismes, définissent une technique préventive hybride tolérante aux pannes qui offre une meilleure gestion d'énergie et augmente donc la durée de vie du réseau. Après

réorganisation du réseau en clusters, la méthode proposée pilote le collecteur mobile pour se déplacer à travers les centres des clusters en prenant le chemin optimal. Le collecteur mobile récupère donc les données depuis les capteurs des clusters visités.

Le principe de KAT-mobility se résume en deux procédures : clustering et optimisation du routage. La figure II-7 illustre le principe de fonctionnement de KAT-mobility.

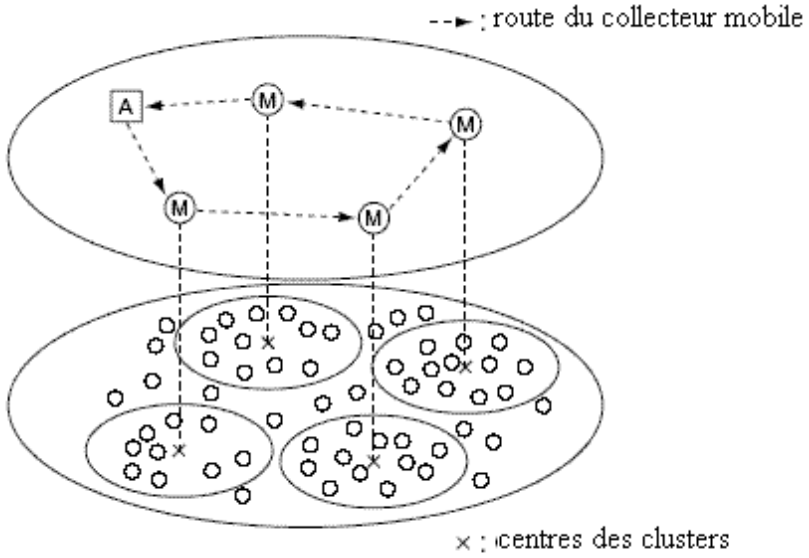

Figure II-7: Fonctionnement de KAT-Mobility

- **Algorithme de clustering :** Cette procédure divise l'ensemble des N capteurs en k clusters C_1, C_2, ..., C_k. Le coût du cluster est évalué par l'erreur approximative entre le collecteur et les nœuds capteurs. Soit $d(x, y_i)$ cette erreur, où x est un capteur et y_i est un collecteur (i = 1, 2, ..., k). d(x, yi) est définie par la distance euclidienne entre le capteur et le collecteur. Le but est donc, d'affecter chaque capteur au clusterhead le plus proche.
- **Optimisation du routage :** cette phase consiste un chemin optimal pour le nœud mobile comme dans le cas du voyageur de commerce (TSP). Ainsi ; un collecteur représente le voyageur, et les centres des clusters définissent les villes. L'optimisation de la route du collecteur mobile consiste à visiter tous les centres des clusters une et une seule fois.

Les résultats de simulation ont montré que KAT-mobility peut fournir une meilleure conservation d'énergie aussi bien qu'une bonne tolérance aux pannes en cas où certains capteurs cessent à fonctionner.

II.5 Conclusion

Dans ce chapitre, nous avons présenté une taxonomie sur la tolérance aux pannes en particulier pour les RCSF qui sont composés de capteurs trop sensibles aux pannes. Puis, nous avons étudié quelques solutions de routage tolérantes aux pannes. Le résultat de cette étude, nous a permis de remarquer que l'approche clustérisée est performante en termes de consommation d'énergie. Dans cette optique, nous avons ajouté à cette approche l'aspect tolérance aux pannes.

Dans le chapitre qui suit, nous proposons d'évaluer quelques approches tolérantes aux pannes sur une plateforme réelle de réseaux de capteurs.

Chapitre III

Développement d'une application tolérante aux pannes pour les RCSF

Chapitre III
Développement d'une application tolérante aux pannes pour les RCSF

III.1 Introduction

Les protocoles de routage existants utilisent souvent le modèle du disque unitaire pour modéliser les communications radio entre les nœuds. Cependant, ce modèle ne peut pas être considéré comme un modèle réaliste puisqu'il suppose que les communications sont toujours fiables tant que la distance séparant les deux nœuds communicants est inférieure ou égale à la portée de transmission. En outre, les capteurs sont généralement déployés dans des zones hostiles et par conséquent ils sont sujets à des pannes à cause de l'épuisement de leurs batteries ou ils subissent des destructions soit accidentelle ou intentionnelle. Ces pannes qui surviennent, peuvent avoir un impact négatif sur le fonctionnement global du réseau. Pour cela, il s'avère nécessaire d'instaurer des mécanismes qui garantissent la fiabilité de livraison des données aux nœuds collecteurs.

Dans ce travail, nous évaluons des schémas de routage dans deux modèles. Le premier est idéal c'est-à-dire qu'il n'y a pas de panne qui peut survenir alors que dans le deuxième il y a une probabilité qu'au moins qu'un capteur cesse de fonctionner. Dans ce cadre nous évaluons deux schémas de routage sur une plateforme réelle de réseaux de capteurs. Le premier est basé sur une architecture plate alors que le deuxième est basé sur une architecture clustérisée.

Le développement de ces applications nécessite des outils logiciels spécifiques aux RCSF tels que TinyOS [21] comme système d'exploitation et NesC [22] comme langage de programmation.

Dans ce chapitre, nous développons une application tolérante aux pannes basée sur une architecture clustérisée avec deux clusterheads : un qui est principal et l'autre est secondaire. Puis nous évaluons ses performances et nous les comparons à une autre non tolérante aux pannes.

III.2 Environnement d'évaluation

Pour concrétiser l'apport de l'agrégation des données dans un environnement tolérant aux pannes, nous avons évalué un réseau de capteurs dans trois scénarios.

- Le premier est une architecture à plat dans laquelle il y a cinq nœuds capteurs qui communiquent directement leurs données captées à la station de base. Puis, nous avons évalué cette architecture dans un modèle probabiliste là où il y a une certaine probabilité qu'un nœud cesse de fonctionner et nous avons varié le taux d'occurrence de pannes pour illustrer l'impact de ce taux sur les performances du réseau en termes de nombre de paquets reçus par la station de base.
- Le deuxième scénario est une architecture clustérisée dans laquelle il y a un nœud agrégateur qui reçoit un ensemble de données des capteurs déployés dans son voisinage. Ce dernier agrège ces données en une seule et la retransmet à la station de base. Puis, nous avons évalué cette architecture dans un environnement dans lequel il y a une probabilité qu'un nœud peut cesser à fonctionner et dans ce cas l'information ne sera pas remontée à la station de base.
- Le troisième scénario est une architecture clustérisée avec deux clusterheads : un qui est principal et l'autre est secondaire. Le deuxième n'envoie l'information que si le premier est en panne.

III.3 Outils logiciels pour les RCSF

Dans cette section, nous présentons les outils utilisés pour la mise en œuvre et l'évaluation des schémas de routage décrits dans la section précédente. Nous commençons tout d'abord par TinyOS, le système d'exploitation conçu pour les dispositifs à ressources limitées en particulier les RCSF. Puis, nous décrivons ensuite le langage de programmation NesC par lequel nous avons développé le code qui reflète les trois scénarios.

III.3.1 Le système d'exploitation "Tinyos"

Les contraintes des RCSF ont motivé l'université de Berkeley et d'autres contributeurs à développer un système d'exploitation destiné aux RCSF afin de faciliter l'implémentation et l'exécution de protocole dédiés à ce type de réseaux.

L'objectif consiste à minimiser la taille du code afin de respecter les contraintes de ressources énergétiques et physiques des nœuds capteurs. Ce système a l'avantage de permettre une programmation simple et puissante tout en gardant la portabilité du code pour les nombreuses plateformes supportées. Il est utilisé par plus de 500 universités et centres de recherche dans le monde vu la caractéristique open-source qu'il détient. Il respecte une architecture basée sur une association de composants et utilise une programmation entièrement réalisée en langage NesC.

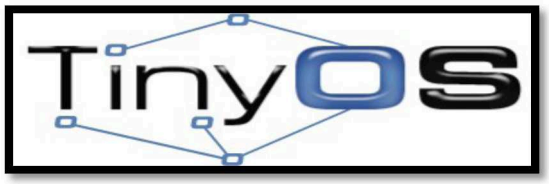

Figure III-1: Sigle de TinyOs

a) Propriétés de TinyOS

TinyOS reste néanmoins le plus répandu pour les RCSFs car il répond aux exigences particulières des applications des RCSF. Il convient alors de mentionner les propriétés qui rendent TinyOS aussi populaire et réputé pour ce genre de réseaux [21]**.**

- Une taille de mémoire réduite.
- Une basse consommation d'énergie.
- Des opérations robustes.
- Applications orientées composants : TinyOS fournit une réserve de composants systèmes utilisables au besoin.
- Programmation orientée évènement : Généralement sur TinyOS, un programme s'exécute suivant le déclenchement des événements. Sinon, les capteurs restent en veille ce qui maximise la durée de vie du réseau.

b) Eléments de Tinyos

TinyOS est construit autour des différents concepts décrits ci-dessous.

- **Les composants :** constitués de :
  - **Frame :** est un espace mémoire de taille fixe permettant au composant de stocker les variables globales et les données qu'il utilise. Il n'en existe qu'un seul par composant.

- **Tâches :** contiennent l'implémentation des fonctions. Elles sont décomposées en deux catégories : les commandes et les évènements.

- **Les interfaces :** représentent le descriptif des fonctions définies dans les tâches.

III.3.2 Le langage NesC

NesC est un langage de programmation orienté composants syntaxiquement proche du langage C. Il est conçu pour la réalisation des systèmes embarqués distribués, en particulier, les RCSF.

NesC s'appuie sur des composants ce qui permet de décomposer une application en modules réutilisable. Il utilise trois abstractions de programmation : interface, module et configuration :

- **Interface** : contient les signatures des commandes (command) et des évènements (event) que le module devra implémenter. Comme TinyOs n'autorise pas les pointeurs de fonctions,NesC propose une alternative : Les interfaces paramétrées. interface SendMessage [uint8 id] permet de créer 256 interfaces de type SendMessage, il suffit alors d'en sélectionner une à l'aide de son identifiant.

- **Module** : permet d'implémenter les composants. Un module est composé de deux blocs, le premier contient les déclarations, le second l'implémentation. Dans le premier bloc, les mots clés <<use>> et <<provide>> permettent de savoir si le module fait appel à une fonction de l'interface (use) ou redéfini son code (provide), <<use nomInterface>> signifie que le module doit fournir l'implémentation des évènements de cette interface, <<providenomInterface>> signifie que le module doit fournir les commandes de cette interface. Le second bloc commence par le mot clé implémentation et se compose des variables privées et de l'implémentation des différentes commandes et évènements. Un module est de la forme suivante :

```
Module nomModule{
Provides {
        //liste des interfaces fournies
        Interface nomInterface }
Uses {
        //liste des interfaces requises
        Interface nomInterface  }
Implémentation {
        //déclaration des variables
        //implémentation des fonctions décrites par les interfaces fournies
} }
```

- **Configuration** : définit le ou les composants qui seront utilisés par l'application (permet de décrire les composants composites). Elle est constituée de modules et/ou d'interfaces et de la déclaration des liaisons entre composants. Elle est de la forme suivante :

Configuration nomConfig{}

Implémentation {
//liste des modules et configurations utilisées
Composants Main, Module$_1$,……..Module$_N$, Config$_1$…….Config$_M$
//description des liaisons
Interfaces requises->Interfaces fournies
}

On distingue les modules et les configurations dans le but de permettre aux concepteurs d'un système de construire des applications rapidement et efficacement. Par exemple, un concepteur peut fournir uniquement une configuration qui relie un ensemble de modules qu'il ne développe pas lui-même. De plus, un autre développeur peut fournir une librairie de modules qui peuvent être utilisés dans la construction d'autres applications.

III.3.3 Langage Java

Java contient les librairies nécessaires au développement d'interfaces homme/machine et étant le seul dont le code exécutable est portable, c'est ce langage qui a été choisi pour bien exploiter l'application au niveau du poste de contrôle par exemple la création d'interfaces graphique.

III.4 Agrégation de données dans un environnement avec pannes

Avant d'évaluer les performances de chaque schéma de routage, nous présentons l'application développée. Dans cette application, nous avons mis en place une architecture clustérisée dans laquelle il y a deux nœuds agrégateurs : un qui est principal et un autre qui est secondaire.

Figure III-2: Architecture clustérisée avec deux agrégateurs

Dans ce cas, les valeurs des températures des trois nœuds membres vont être affichées ainsi que la valeur agrégée par l'agrégateur 2 puisque l'agrégateur 1 a cessé de fonctionner.

Valeur agrégée = (29°+28°+33°) /3 qui donne 30° comme le montre la figure III-2.

III.5 Evaluation des performances

Dans cette section, nous mettons en place les schémas de routage décrits dans la section 2 et nous les évaluons dans deux modèles : un qui est idéal et un autre dans lequel il pourrait y avoir l'occurrence de pannes.

III.5.1 Premier scénario : architecture plate

Dans cette architecture, nous mettons en place l'architecture plate dans deux modèles : modèle idéal et modèle avec pannes. Cette architecture s'articule autour de cinq nœuds capteurs qui collectent la température et l'envoient à la station de base comme s'est illustré par les figure III-3 et III-4.

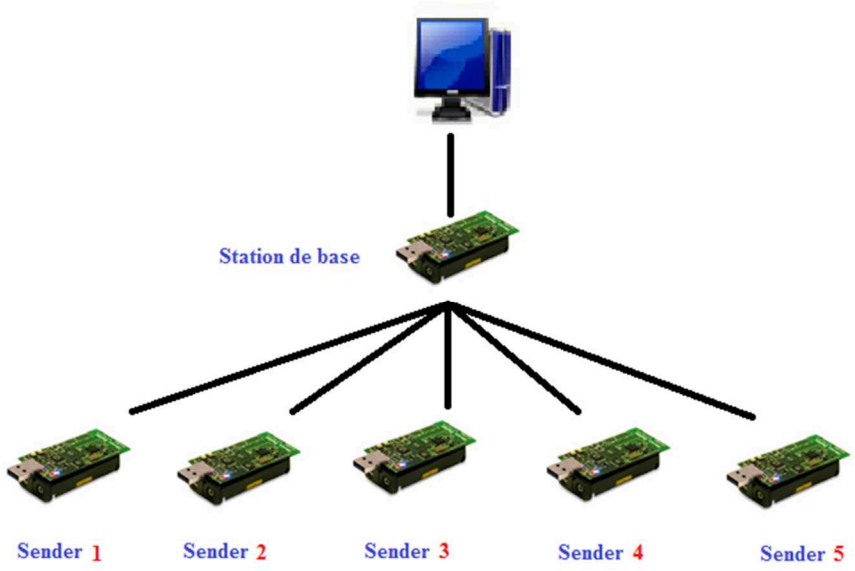

Figure III-3: Architecture plate sans pannes

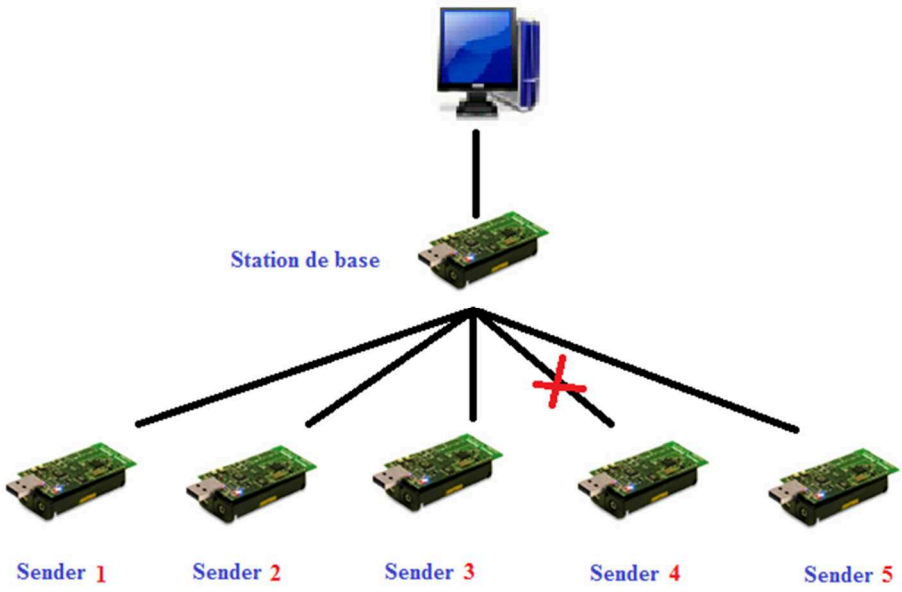

Figure III-4: Architecture plate avec pannes

Pour illustrer l'impact de la panne sur les performances du réseau, nous avons instauré un modèle probabiliste dans lequel nous avons fait varier le taux de pannes de 10%, 20%, 30%, 40% et 50%. Puis, nous avons programmé les capteurs de telle sorte que chacun deux envoie une température chaque seconde.

Le tableau III-1 et la figure III-5 reflètent les résultats de ces évaluations sur une plateforme réelle.

Tableau III-1 : Nombre de paquets reçus en fonction du taux de pannes

Temps	**5 minutes**				
Taux de pannes	10%	20%	30%	40%	50%
Nombre de paquets reçus	1065	995	944	901	872

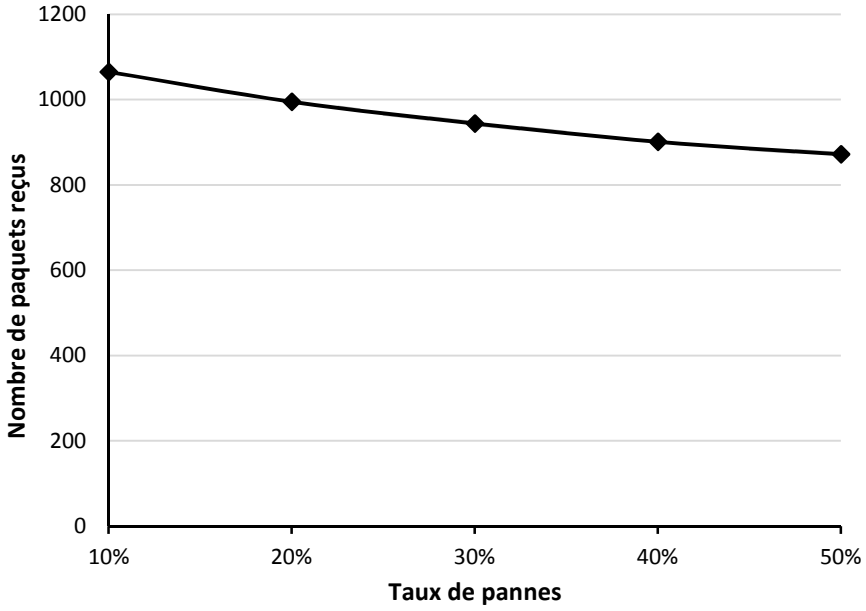

Figure III-5 : Nombre de paquets reçus en fonction des taux de pannes

D'après la figure III-5, nous remarquons que le nombre de paquets diminue quand le taux de pannes augmente

III.5.2 Deuxième scénario : un agrégateur et trois nœuds membres

Dans ce contexte, nous avons évalué cette architecture dans deux modèles : modèle sans pannes et modèle avec pannes. Les figures III-6 et III-7 représentent respectivement ces deux modèles.

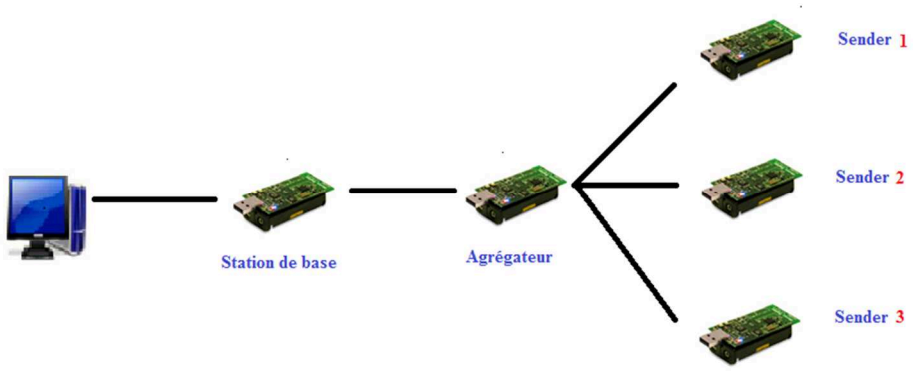

Figure III-6: Architecture avec un agrégateur sans pannes

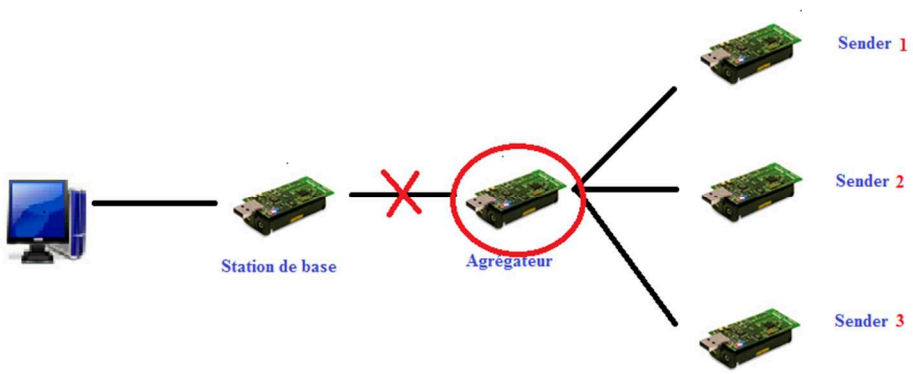

Figure III-7: Architecture à un agrégateur avec pannes

Tableau III-2 : Résultats du deuxième scénario

Temps	**5 minutes**				
Taux de pannes	10%	20%	30%	40%	50%
Nombre de paquets reçus	765	740	678	629	617

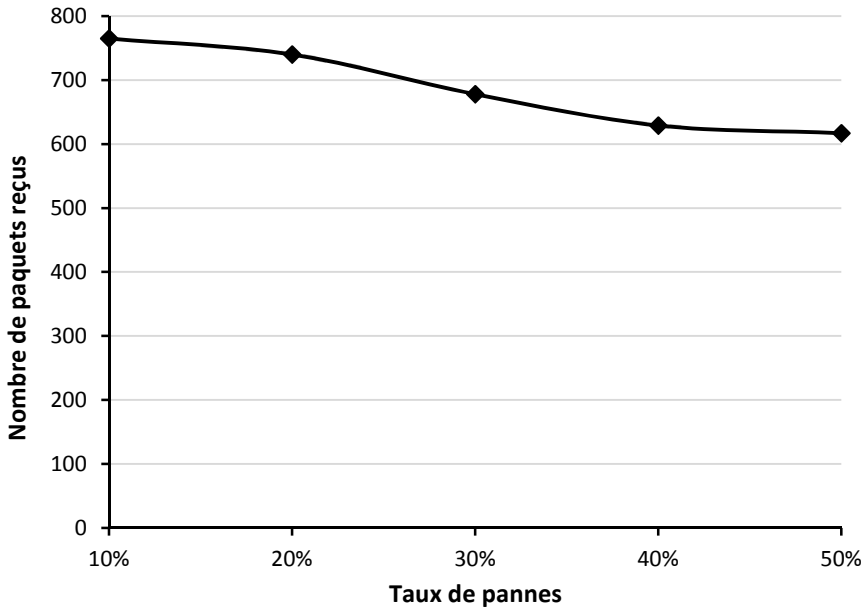

Figure III-8 : Nombre de paquets reçus dans le deuxième scénario

III.5.3 Troisième scénario : Deux nœuds agrégateurs et trois nœuds membres

Dans cette architecture, nous avons mis en place une architecture clustérisée telle qu'au niveau du cluster il y a deux nœuds agrégateurs : un qui est principal et un autre qui est secondaire. En outre, dans cette architecture tant que l'agrégateur principal est fonctionnel, l'agrégateur n'envoie aucune information à la station de base.

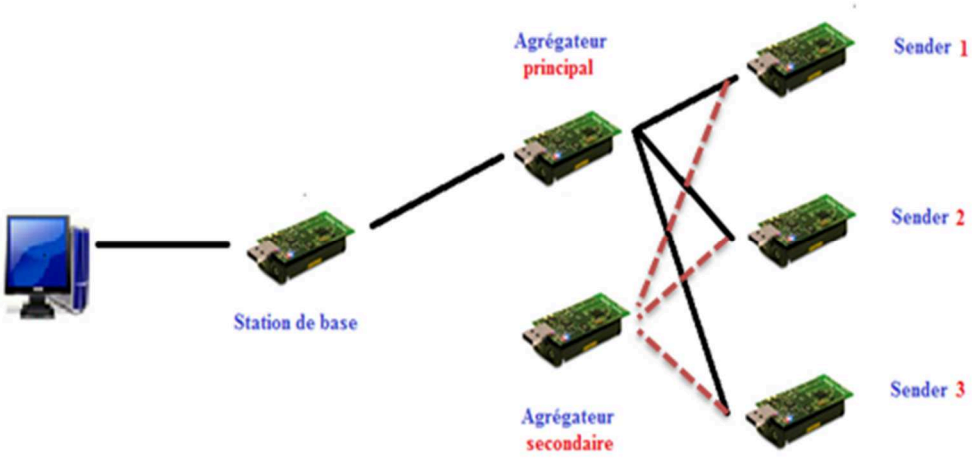

Figure III-9: Architecture avec deux agrégateurs sans pannes

La figure III-9 illustre le cas de panne de l'agrégateur principal et dans ce de figure c'est l'agrégateur adjoint qui agrège l'information et l'envoie à la station de base.

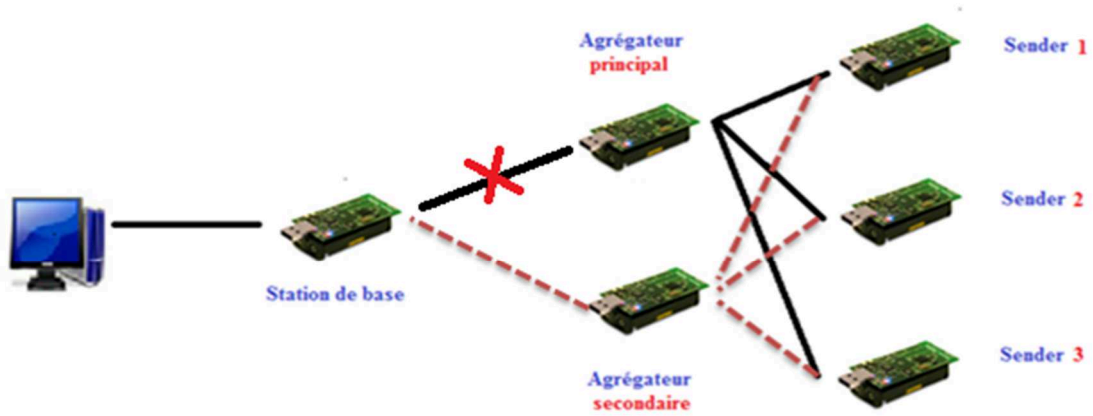

Figure III-10: Architecture avec deux agrégateurs avec pannes

Le tableau III-3 représente les résultats obtenus après cinq minutes d'exécution dans un environnement avec pannes.

Tableau III-3 : Résultats du troisième scénario

Temps	5 minutes				
Taux d'erreurs	10%	20%	30%	40%	50%
Nombre de paquets reçus	801	763	694	642	632

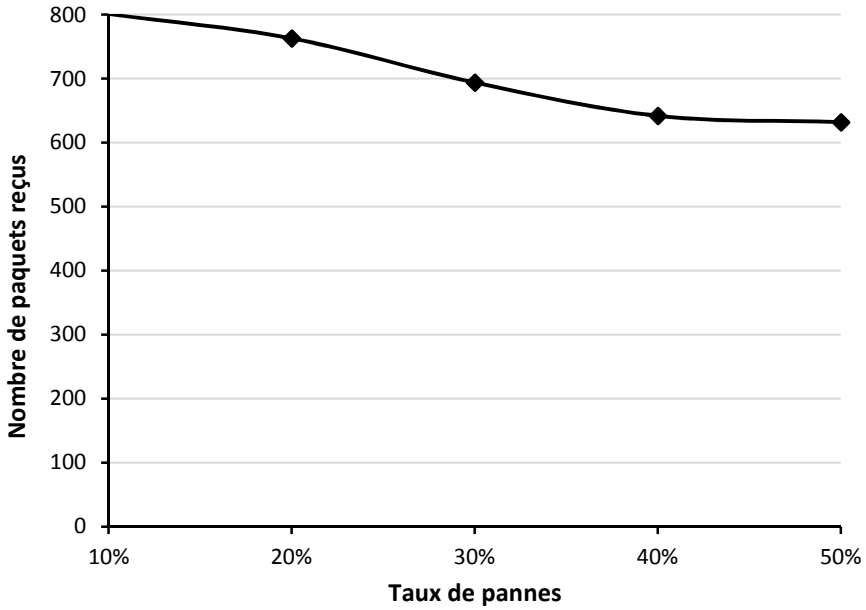

Figure III-11 : Résultats du troisième scénario

III.6 Conclusion

Dans ce chapitre, nous avons développé une application tolérante aux pannes dans une architecture clustérisée. Puis nous avons évalué les performances de trois schémas de rouge dans deux modèles : un modèle idéal sans pannes et un autre qui est probabiliste dans lequel il y a une probabilité d'occurrence de pannes.

Les résultats obtenus ont montré que quand le taux de pannes est élevé le nombre de paquets reçus au niveau de la station de base sera diminué. Ceci explicite que les pannes peuvent un impact négatif sur le fonctionnement d'un réseau de capteurs.

Conclusion générale

Conclusion générale

Les capteurs sont sujets à des pannes et ces pannes ont un impact négatif sur le bon fonctionnement des RCSF. Dans ce contexte, nous avons développé une application tolérante aux pannes de telle sorte que le réseau peut continuer à fonctionner malgré l'occurrence des éventuelles pannes.

Ce projet nous a permis d'acquérir des connaissances en programmation évènementielle. Il nous a aussi fait découvrir un nouveau langage de programmation, le NesC ainsi que la plateforme de programmation adéquate qui est TinyOs.

En perspectives, nous proposons d'évaluer d'autres protocoles dans un environnement réaliste et proposer des versions améliorées qui s'adaptent à ce type d'environnement.

Bibliographie

Références bibliographiques

[1] L.Quin and T.Kunz. On-demand routing in MANETs : The impact of a realistic physical layer model. In proceedings of the International Conference on Ad hoc, Mobile, and Wireless Networks, pp.37-48, Montreal, Canada, 2003.

[2] Olivier Français, "Capteurs et électronique associée", Chapitre (type,lieu), 2000.

[3] Vernon S. Somerset, "Intelligent and Biosensors, Edited by Vernon S. Somerset", Intech, January 2010.

[4] YOUSEF Yaser, "Routage pour la Gestion de l'Energie dans les Réseaux de Capteurs Sans Fil", Thèse de Doctorat, Université de haute alsace, Juillet 2010.

[5] Dima HAMDAN, "Détection et diagnostic des fautes dans des systèmes à base de réseaux de capteurs sans fil", Thèse de Doctorat, Université de Grenoble, Février 2013.

[6] Ali MAHMOUD, Annette BOHM and Magnus JONSON. "Wireless sensor networks for surveillance applications - A comparative survey of MAC protocols". In Proceedings of the 4th International Conference on Wireless and Mobile Communications (ICWMC'08), pp399–403, Washington, DC, USA, 2008.

[7] B. Selic. «fault tolerance techniques for distributed systems». http://www.ibm.com/developerworks/rational/library/114.html, 2004.

[8] R. Garg and A. K. Singh, "Fault Tolerance in Grid Computing: State of the Art and open issues", International journal of Computer Science & Engineering Survey, Vol. 2, No. 1, February 2011.

[9] Ajay, N.Tarasia, S. Dash, S.Ray, ARSwain, "Une erreur dynamique tolérant protocole de routage pour prolonger la durée de vie des réseaux de capteurs sans fil", Journal (IJCSIT), Vol. 2 (2), pp727-734, 2011.

[10] Ajay, N.Tarasia, S. Dash, S.Ray, ARSwain, "Protocole de routage tolérant aux fautes multi-niveaux avec des horaires du sommeil (FMS) pour les réseaux de capteurs sans fil", European Journal of Scientific Research,, Vol. 55(1) : pp97-108, 2011.

[11] K. Kulothungan, J. Angel Arul Jothi, A. Kannan «Une erreur de protocole de routage adaptatif tolérant pour les réseaux de capteurs sans fil»European Journal of Scientific Research ISSN 1450-216X N ° 1 Vol.60 (2011), pp 19-32.

[12] Guowei Wu, ChiLin, Feng Xia, Lin Yao, il Zhang et Liu Bing «Saut dynamique en temps réel Fault-Tolerant protocole de routage pour les réseaux de capteurs sans

fil» de la Fondation nationale des sciences naturelles de Chine par la concession numéro60703101 et n ° 60903153 (2010).

[13] Zamree Che-Aron, Wajdi Al-Khateeb, et Farhat Anwar «Le Renforcement de tolérance de panne Mécanisme de protocole de routage AODV pour le réseau de capteurs sans fil» IJCSNS International Journal of Computer Science et de sécurité réseau, Vol.10 No.6, Juin 2010.

[14] Benhamida, F. Z, and Challal, Y., "Fault Tolerant Directed Diffusion for Wireless Sensor Networks", International Conference on Availability, Reliability, and Security, 2010 (ARES '10), pp112-118, Krakow, Poland.

[15] Azzedine Boukerche et al., "A Fast and Reliable Protocol for Wireless Sensor Networks in Critical Conditions Monitoring Applications", International Conference (MSWiM'04), Canada, 2004.

[16] Debraj De et al., "EAR: An Energy and Activity-Aware Routing Protocol for Wireless Sensor Networks in Smart Environments", Computer Journal 55 (12): pp1492-1506, 2012.

[17] Azzedine Boukerche et al. "A new energy efficient and fault-tolerant protocol for data propagation in smart dust networks using varying transmission range", Journal of computer communications, Vol. 29 no. 4, pp477-489, 2006.

[18] A. Boukerche et al., "Fault-tolerant wireless sensor network routing protocols for the supervision of context-aware physical environments", Journal of Parallel and Distributed Computing, Vol. 66 no. 4, pp586-599, 2006.

[19] Fei Dai and Jie Wu, "On constructing k-connected k-dominating set in wireless ad hoc and sensor networks", Journal of Parallel and Distributed Computing, Vol. 66 no. 7, pp947-958, July 2006.

[20] NAKAYAMA Hidehisa et al., "Fault-resilient sensing in wireless sensor networks", Journal of Computer communications, vol. 30, n° 11-12, pp. 2375-2384, 2007.

[21] Tinyos. http:/www.tinyos.net/, 2010.

[22] NesC: A Programming Language for Deeply Networked Systems. http://nescc.sourceforge.net/

Annexe : Installation de TinyOS-2.1.2

Il y a deux façons de faire une installation propre de TinyOs. La première façon est d'installer une VM qui a une installation complète de TinyOs. La deuxième façon est d'installer TinyOs sur votre système d'exploitation hôte. Lors de l'installation sur un système d'exploitation hôte, vous pouvez soit utiliser un paquet Debian ou installer manuellement avec RPM. On a fait une installation sur un Os avec un paquet Debian.

L'installation se fait en deux étapes sur votre système d'exploitation hôte avec les paquets Debian : [20]

- Retirez tout ancien référentiel TinyOs du fichier : / etc / apt / sources.list et ajoutez la ligne suivante : **deb** http://tinyos.stanford.edu/tinyos/dists/ubuntu **lucid main.**
- Mettez à jour votre cache de dépôts :

sudo apt-get update

Puis exécutez la commande suivante pour installer la dernière version de TinyOs et tous ses outils pris en charge :

sudo apt-get install tinyos

Cela vous donnera probablement un message vous invitant à choisir entre les deux versions disponibles. Un exemple à exécuter ensuite est :

Suivante à votre fichier ~ /. Bashrc ou ~ /.profil dans votre répertoire home afin de mettre en place l'environnement pour le développement TinyOS lors de la connexion.

#Sourcing the tinyos environment variable setup script
source /opt/tinyos-2.1.2/tinyos.sh

Si vous exécutez généralement TinyOS à partir de CVS et nécessitent seulement l'installation de **toolchain**, vous pouvez installer le paquet **tinyos-required** au lieu de**tinyos**.

Pour l'instant il est préférable de supprimer tous les anciens paquets **TinyOs** avant d'installer les nouveaux. Aussi, vous avez utilisé le référentiel TinyOs Debian dans le passé, garder à l'esprit que tous les outils ont été mis à jour pour TinyOs-2.1.2, mais toujours travailler avec toutes les anciennes versions de TinyOs.

Résumé

Dans ce projet, nous avons visé les schémas de routage tolérants aux pour garantir la fiabilité de livraison des données à la station de base. Dans cette optique, nous avons mis en place trois scénarios de routage et nous avons évalué leurs performances sur une plateforme réelle de RCSF dans deux modèles : un qui est sans pannes et un autre il y aurait l'occurrence d'éventuelles pannes.

Les résultats obtenus ont montré que les pannes peuvent avoir un impact négatif sur le bon fonctionnement du réseau si on n'instaure pas des mécanismes de tolérance aux pannes.

La mise en place de cette application exige des outils matériels et logiciels bien spécifiques puisque les capteurs sont des composants à ressources limitées. Pour cela, nous avons utilisé le langage NesC qui fait minimiser l'utilisation de la mémoire et TinyOs qui est un système d'exploitation léger.

Mots clés : Réseaux de capteurs, tolérance aux pannes, agrégateurs, NesC, TinyOs.

Abstract

ملخص.: