N° d'ordre: 20/2012-M/INF

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumediene
Faculté d'Electronique et d'Informatique
USTHB/Alger

Mémoire

Présenté pour l'obtention du diplôme de MAGISTER

En: INFORMATIQUE

Spécialité: Informatique Mobile

Par : M. Loucif AMIROUCHE

Sujet:

Tolérance aux Défaillances des Actionneurs dans les Réseaux de Capteurs et d'Actionneurs.

Soutenu publiquement le 13/07/2010, devant le jury composé de :

Dr	**Mahfoud BENCHAIBA**	**Maitre de Conférences/A à l'USTHB**	**Président**
Prof	**Nadjib BADACHE**	**Professeur à l'USTHB**	**Directeur de Thèse**
Dr	**Djamel DJENOURI**	**Maitre de Recherche/A au CERIST**	**Co-Encadreur**
Dr	**Omar NOUALI**	**Maitre de Recherche/A au CERIST**	**Examinateur**

Remerciements

En ouverture, je tiens à remercier en premier lieu le Professeur Nadjib BADACHE et Docteur Djamel DJENOURI, mes directeurs de thèse pour leur encouragement, leur disponibilité, leurs idées, leurs conseils et leur sympathie qui m'ont permis de mener à bien cette thèse.

Je tiens à dire toute ma gratitude au Docteur Djamel DJENOURI ! Je pourrai écrire un cinquième chapitre rien que sur ce jeune et talentueux Docteur. Merci! Pour m'avoir encadré au sens propre du terme, puis pour m'avoir fait confiance en me confiant ce sujet de thèse et pour m'avoir accompagné, encouragé et conseillé au cours de ces années tant sur le plan technique que humain.

Je tiens à remercier très chaleureusement les membres du jury. J'aimerais aussi souligner la gentillesse particulièrement touchante de Dr BENCHAIBA Mahfoud Maitre de Conférences Classe A à l'USTHB qui a présidé le jury de ma thèse. Merci au Dr NOUALI Omar, Maitre de recherche Class A, CERIST, pour m'avoir fait l'honneur d'accepter de rapporter ce travail.

J'adresse également mes sincères remerciements à ma famille; ma mère et mes deux sœurs pour m'avoir aidé à surmonter tous les obstacles et à me forger à travers les difficultés vécues durant toute cette période de travail. C'est donc avec le plus grand plaisir que je remercie ma très chère maman qui n'a jamais cessé de me demander mon état d'avancement chaque week-end.

À présent, je m'arrête pour saluer tous les acteurs de mon quotidien au CERIST, à l'USTHB et à ELIT, c'est l'occasion de remercier toutes les personnes que j'ai pu croiser durant ces années et qui ont toutes, à leur manière, embelli ces années de thèse.

Un grand merci à tous les membres du laboratoire réseaux de capteur au sein de CERIST qui m'ont procuré une ambiance chaleureuse pour effectuer mon travail. Je tiens à remercier particulièrement Mr Abdelraouf OUADJAOUT pour les précieux conseils qui m'a prodigué sur la partie simulation.

Mes derniers remerciements vont à Cheraz, qui m'accompagne avec tant de bonheur depuis un bon bout de temps maintenant. Un ange qui me fait oublier la fatigue et le stress de tous les jours. Merci pour ta patience et ton aide à nulle autre pareille. Les derniers mois de thèse furent, à tes côtés, un ravissement et une joie que nous continuons à partager.

Enfin et avant tout, le grand et le vrai merci à Dieu qui m'a donnée la force et la vie pour accomplir cette tâche.

À vous tous, merci !

Résumé

La nécessité des WSANs s'est fait sentir par ses divers domaines d'application, qu'ils soient civils, médical ou militaires. De plus, les enjeux économiques qu'il représente et la qualité de service qu'il permet d'obtenir en précision et temps de réponse, font de ce type de réseau un domaine de recherche très attractif.

Néanmoins, cette classe de réseaux émergente soulève un certain nombre de défis, tel que la tolérance aux défaillances des actionneurs déployés avec un nombre limité et devant parfois être mobiles pour couvrir une région plus grande. Ces actionneurs, en plus des pannes traditionnelles de batterie et d'interface radio, sont sujets à des pannes mécaniques sur l'unité d'actuation.

Peu de travaux ont été réalisés pour répondre au problème de la tolérance aux défaillances des actionneurs dans les WSANs. Dans [OHE07], Les auteurs on utilisés un model Multi Actionneurs Multi Capteur avec une coordination semi-passive (SPC). Cette dernière ce base principalement sur l'élection d'un actionneur primaire unique qui prend la décision de la tache à exécuter suivant les données captées transmises par les capteurs et les autres actionneurs backup. En cas de défaillance de l'actionneur primaire, un algorithme de consensus est exécuté pour choisir un autre actionneur primaire parmi les backups. Cette technique permet d'assurer la tolérance aux défaillances des actionneurs, mais avec une certaine latence, le temps de choisir un remplaçant à l'actionneur primaire. Par ailleurs, nous avons constaté que les techniques de redondance et de réplication utilisées dans WSN ne sont pas adaptées pour tolérer les pannes des actionneurs : la première exige un grand nombre d'actionneurs, et la seconde augmente la consommation d'énergie des capteurs qui font transiter la donnée dupliquée entre les différents actionneurs.

Nous avons proposé un protocole de clustering rapide et local qui permet d'obtenir des clusters autoréparables ***CA*** de taille égales où chaque cluster contient deux actionneurs CHi et CHj. La gestion d'un cluster *Ci* en interne incombe au *CHi.* Dans le cas de panne d'un *CHi,* le *CHj* voisin membre d'un même CA peut le remplacer dans toutes ses taches de décision et d'actuation. Nous avons aussi proposé un protocole de routage adapté qui prend en compte les contraintes de latence et de consommation d'énergie.

Nous avons simulé notre protocole sous TOSSIM et nous l'avons comparé à des protocoles existant pour montrer son efficacité en termes de taux de réussite et de latence de temps d'exécution.

Mots clés : Réseau de capteurs et d'actionneurs, Tolérance aux défaillances, Clustering, QoS, Routage dynamique, Tinyos, TOSSIM.

Sommaire

Introduction

L'avancée technologique dans le domaine de la miniaturisation des composants électroniques et la communication sans fil a permis l'apparition d'un nouveau type de réseaux appelé réseau de capteurs ou Wireless Sensor Networks (WSNs). Ce dernier a une très grande variété d'applications dans le domaine civil, médical ou encore militaire.

En raison de ses multiples domaines d'applications, ce type de réseau a vu des travaux de recherche se multiplier sur des axes pointus comme la communication, l'optimisation de la consommation d'énergie et la tolérance aux défaillances. Cependant, les WSNs sont limité uniquement à capter les grandeurs physiques d'un environnement (chaleur, humidité, vibration etc.…).

L'intérêt grandissant pour une interaction intelligente avec l'environnement a engendré l'apparition d'une autre classe de réseaux, appelée : réseaux de capteurs et d'actionneurs (WSANs ou SANet), capables de capter des grandeurs physiques comme dans les WSNs, mais aussi, de réagir dans l'environnement en question, en réponse aux données captées.

Les WSANs soulèvent de nouveaux challenges pour la tolérance aux défaillances. Ces réseaux sont sujets aux fautes à cause de la nature partagée du médium sans fil. De plus, les pannes des nœuds à cause d'un défaut de fabrication ou l'épuisement de l'énergie sont courantes. Ajouter à cela, la réaction des actionneurs dans l'environnement physique qui effectue souvent des taches mécaniques très sensibles aux pannes, rend la tolérance aux défaillances encore plus critiques pour ce type de réseaux.

En effet, les WSANs trouvent leur place dans un grand nombre de domaines d'applications. Chacun de ces domaines, impose des contraintes supplémentaires, telles la communication à temps réel et la tolérance aux défaillances « *Fault Tolerance »* (**FT**). Un exemple est la détection et l'extinction des incendies de forêts, où le temps de latence entre le captage de la donnée et l'intervention de l'actionneur doit être minimal pour éviter la propagation du feu, d'où le transfert des données captées en temps réel, ainsi que la tolérance aux défaillances des actionneurs sont cruciaux pour permettre à ces derniers de réagir efficacement à l'événement capté.

Le souci d'automatisation d'un tel système, et ainsi, le rendre fonctionnel quelque soient les défaillances auxquelles il est confronté, exige une tolérance à tous les types de défaillances, et sur tous les nœuds présents dans le réseau.

Peu de travaux ont été réalisés pour répondre au problème de la tolérance aux défaillances des actionneurs dans les WSANs. Dans [OHE07], Les auteurs on utilisés un model Multi Actionneurs Multi Capteur avec une coordination semi-passive (SPC). Cette dernière ce base principalement sur l'élection d'un actionneur primaire unique qui prend la décision de la tache à exécuter suivant les données captées transmis par les capteurs et les autres actionneurs backup. En cas de défaillance de l'actionneur primaire, un algorithme de consensus est exécuté pour choisir un autre actionneur primaire parmi les backups. Cette technique permet d'assurer la tolérance aux défaillances des actionneurs, mais avec une certaine latence (temps de choisir un remplaçant à l'actionneur primaire). De plus, en cas de panne de l'interface radio de l'actionneur primaire, les backups ne peuvent pas détecter la panne.

Nous avons proposé une nouvelle technique de clustering qui permet de découper le réseau en plusieurs clusters autoréparables. Nous avons aussi proposé un protocole de routage adapté pour faire face aux pannes des actionneurs, mais aussi des capteurs tout en prenant en compte les contraintes de latence et de consommation d'énergie. Nous avons simulé notre protocole sous TOSSIM et nous l'avons comparé à des protocoles existant pour montrer l'efficacité de notre protocole en termes de taux de réussite et de latence de temps d'exécution.

Dans ce qu'il va suivre, nous allons introduire la notion de réseaux de capteur et d'actionneur. Dans le Chapitre II, nous définirons la tolérance aux défaillances. Nous étudierons aussi, les différentes techniques utilisées pour la détection et la prise en charge des défaillances dans les WSANs. Dans le chapitre III, nous présenterons notre algorithme de clustering et notre protocole de routage adapté qui permet la communication à temps réel. Le dernier chapitre serra dédier a la simulation de notre protocole sous TOSSIM dans le but de le comparé aux travaux existants dans le même domaine.

Chapitre I : Les réseaux de capteurs et d'actionneurs.

1. Introduction :

L'avancée technologique dans le domaine de la miniaturisation des composants électroniques et la communication sans fil a permis l'apparition d'un nouveau type de réseau appelé réseau de capteurs ou Wireless Sensor Networks (WSNs). Ce dernier a une très grande variété d'applications dans le domaine civil, médical ou encore militaire. Pour des multiples raisons ce type de réseau a vu des travaux de recherches se multiplier sur des axes pointus comme la communication, l'optimisation d'énergie et la tolérance aux défaillances. Cependant, le type d'actions réalisées par les WSNs est limité uniquement à capter les grandeurs physiques d'un environnement (chaleur, humidité, vibration etc.…).

L'intérêt grandissant pour une interaction intelligente avec l'environnement a engendré l'apparition d'une autre classe de réseaux, appelée : réseaux de capteurs et d'actionneurs (WSANs ou SANet), capables de capter des grandeurs physiques, mais aussi de réagir dans l'environnement en question, par rapport aux données captées.

Le domaine des WSANs attire l'attention de la communauté de chercheurs par la richesse des défis théoriques et pratiques qu'il a engendrés. Ce réseau à grande échelle peut capter les données d'un environnement, traiter ces données et réagir en conséquence.

Dans ce qui va suivre, nous allons introduire la notion de réseaux de capteur et d'actionneur en définissant tout d'abord, les différents nœuds du réseau, mais aussi, les architectures utilisées pour ce type de réseaux. Par la suite, nous présenterons les différents domaines d'applications où nous pourrons tirer profit de déploiement d'un WSAN.

2. Micro-capteur :

2.1. Présentation :

Les progrès réalisés ces dernières décennies dans les domaines de la microélectronique et des technologies de communication sans fil, ont permis de produire avec un coût raisonnable des composants de quelques millimètres cubes de volume, un peu plus grands qu'une pièce de monnaie **-Figure 1-.** Ces derniers, appelés micro-capteurs, intègrent : une unité de captage chargée de capter des grandeurs physiques (chaleur, humidité, vibrations etc.…) et de les transformer en grandeurs numériques ; une unité de traitement informatique et de stockage de données ; et un module de transmission sans fil.

Deux (02) approches sont à signaler dans la construction d'un micro-capteur :

Micro-Capteurs intégrés : Ils intègrent l'unité de captage directement dans le micro-capteur. Cette approche est plus économique et plus robuste, elle se traduit aussi par une diminution visible de la taille du micro-capteur Figure 1.

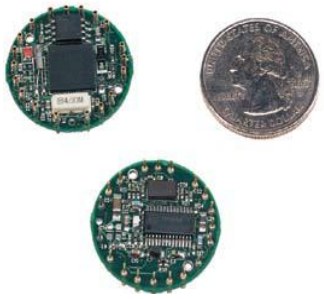

Figure 1: Micro-capteurs intégrés (Mica2dot)

Micro-Capteurs de carte (Sensor Board) : L'unité de captage est connectable au Micro-Capteur à travers des bus d'extensions Figure 2. Cette approche est plus à la demande de l'utilisateur qui peut connecter des capteurs de différents types suivant le domaine d'application sur le même micro-capteur [CRO].

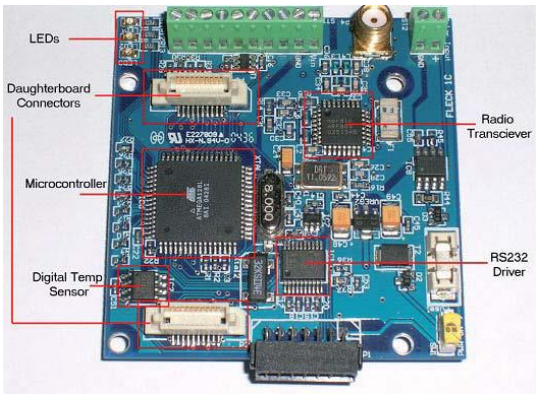

Figure 2: Micro-capteurs carte (Fleck1)

2.2. Caractéristiques physiques d'un micro-capteur :

2.2.1. Petite taille :

La taille d'un micro-capteur varie selon la composition entre 1cm^3 dans le cas d'un micro capteur intégré Figure 1, et 4cm*4cm dans le cas d'un micro-capteur à carte avec des extensions possibles. Sa taille relativement petite lui permet de se loger dans beaucoup d'endroits avec discrétion.

2.2.2. Sources d'énergie très limitées :

L'énergie constitue le principal handicape des micro-capteurs. La taille de la batterie détermine la taille du capteur. Même avec une batterie relativement grande (2pile AAA) l'optimisation de l'énergie surtout en communication, est de mise.

2.2.3. Taux de transfert limité :

Un faible débit de données n'est pas handicapant pour un réseau de capteurs où les fréquences de transmission ne sont pas importantes. Le taux de transfert dans les micro-capteurs est limité à 250Kbps, taux théorique pour le protocole ZigBee sur une fréquence de 2.4 GHz. Figure 3

Protocole	Zigbee	Bluetooth	Wi-Fi
IEEE	802.15.4	802.15.1	802.11a/b/g
Besoins mémoire	4-32 Kb	250 Kb +	1 Mb +
Autonomie avec pile	Années	Jours	Heures
Nombre de nœuds	65 000+	7	32
Vitesse de transfert	250 Kb/s	1 Mb/s	11-54-108 Mb/s
Portée	100 m	10-100 m	300

Figure 3: Tableau comparatif des standards IEEE

2.2.4. Faible coût de fabrication :

Le faible coût de production constitue un avantage considérable pour les micro-capteurs, ce qui permettra de généraliser et diversifier leur utilisation, et pourquoi pas, palier au défaut d'énergie par le faible coût de production.

2.2.5. Autonome et adaptative :

Un micro-capteur intégré est complètement autonome, il peut effectuer toutes les tâches qui lui sont affectées (capter, communiquer…) sans avoir besoin d'un composant tiers. En revanche, il est moins adaptatif qu'un micro-capteur à carte qui est très adaptatif et peut recevoir plusieurs capteurs pour différentes grandeurs physiques.

2.3. Architecture matérielle d'un micro-capteur :

Un nœud capteur contient quatre (04) unités de base : l'unité de captage, l'unité de traitement, l'unité de transmission, et l'unité de contrôle d'énergie [AkK04]. Il peut contenir également, suivant son domaine d'application, des modules supplémentaires tels qu'un système de localisation (GPS), ou bien un système générateur d'énergie (cellule solaire). On peut même trouver des micro-capteurs, un peu plus volumineux, dotés d'un système mobilisateur chargé de déplacer le micro-capteur en cas de nécessité Figure 4.

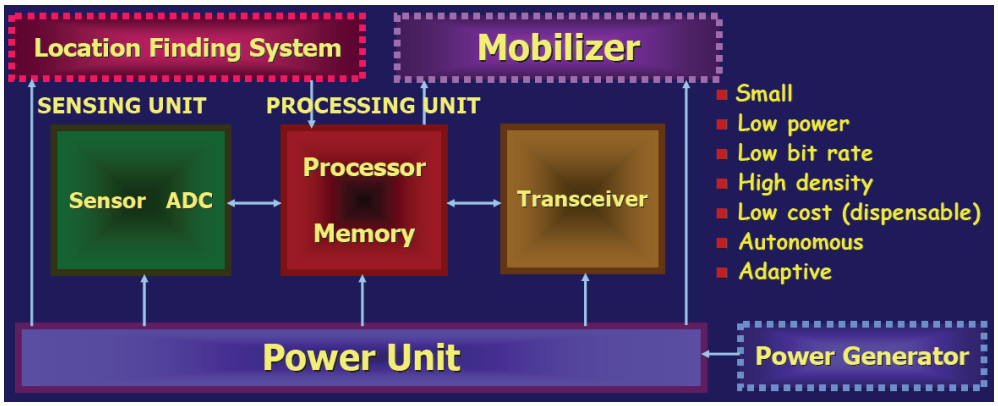

Figure 4 : Architecture matérielle d'un micro-capteur.

2.4. Types de capteur :

Différents types de capteurs peuvent se greffer sur un micro-capteur pour étendre ses capacités tels :

Capteurs de distance : ultrason, micro-onde, triangulation…

Capteurs de lumière : photodiode, capteur photographique…

Capteurs de sons : microphone, hydrophone…

D'autres capteurs de température, pression, débit ou encore d'inertie (pour le calcul de l'accélération) sont tous des capteurs qui peuvent collaborer en réseau dans divers domaines d'applications.

3. Actionneur

3.1. Présentation :

Dans les WSANs les phénomènes de capture grandeur physique et de réaction dans l'environnement sont réalisés par les micro-capteurs et les actionneurs, respectivement. De ce fait, les actionneurs sont des nœuds richement équipés avec une plus grande capacité de calcul et de communication à longue distance, tout en gardant une durée de vie de batterie plus longue. Ainsi, un actionneur est beaucoup plus cher qu'un micro-capteur, d'où le déploiement des micro-capteurs est beaucoup plus important que le nombre d'actionneurs.

Deux (02) notions sont à éclaircir :

Actuator: Périphérique qui peut convertir un signal de contrôle électrique en une action physique. Il constitue le mécanisme avec lequel l'agent réagit sur l'environnement physique.

Actor ou Actionneur : En plus d'être capable de réagir dans un environnement en utilisant un ou plusieurs Actuator, il est aussi une entité réseau qui peut effectuer des opérations de communication, c'est-à-dire, recevoir et transmettre des données [MPA06].

Un exemple d'actionneur est Robotic Mule Figure 5**.** Ce robot de guerre peut détecter et marquer les mines. Une plateforme d'hélicoptère Figure 6, qui peut se coordonner avec des micro-capteurs en surface et avoir ainsi beaucoup de fonctions telles que l'extinction des feux ou la dispersion de gaz avec une grande précision [AkK04].

Figure 5: Robotic Mule.

Figure 6: Aerial mapping helicopter.

3.2. Architecture matérielle d'un actionneur :

Un actionneur contient en plus des unités de transmission, de contrôle d'énergie et de traitement qui sont bien plus puissantes que sur les micro-capteurs, une unité de décision ou le contrôleur et une unité de réaction (actuation unit)Figure 7.

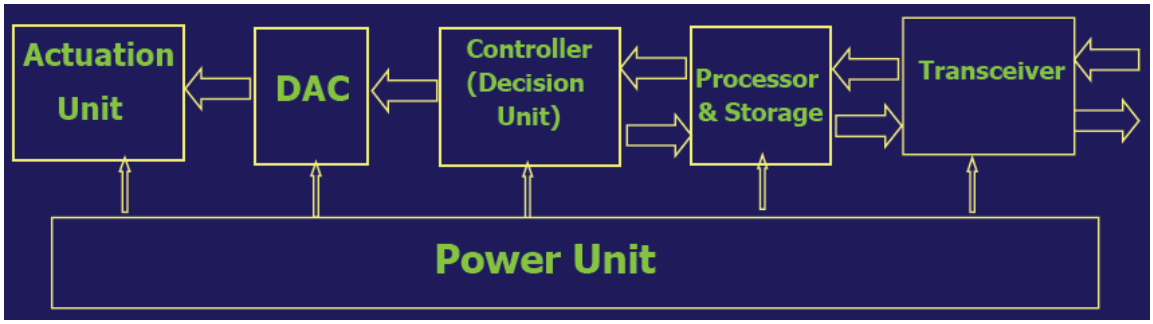

Figure 7: Architecture matérielle d'un actionneur

4. Architectures d'un Réseau de capteur et d'actionneur :

Un micro-capteur détecte les phénomènes physiques et les transmet aux actionneurs qui traitent les données et déclenchent l'action appropriée, ou route les données vers le nœud puits (Sink) qui retourne les commandes à exécuter aux actionneurs. Cette approche est appelée Semi- automatique architecture, par opposition à l'autre cas de figure nommé architecture automatique Figure 8. Dans ce qui suit nous allons détailler ces deux (02) cas.

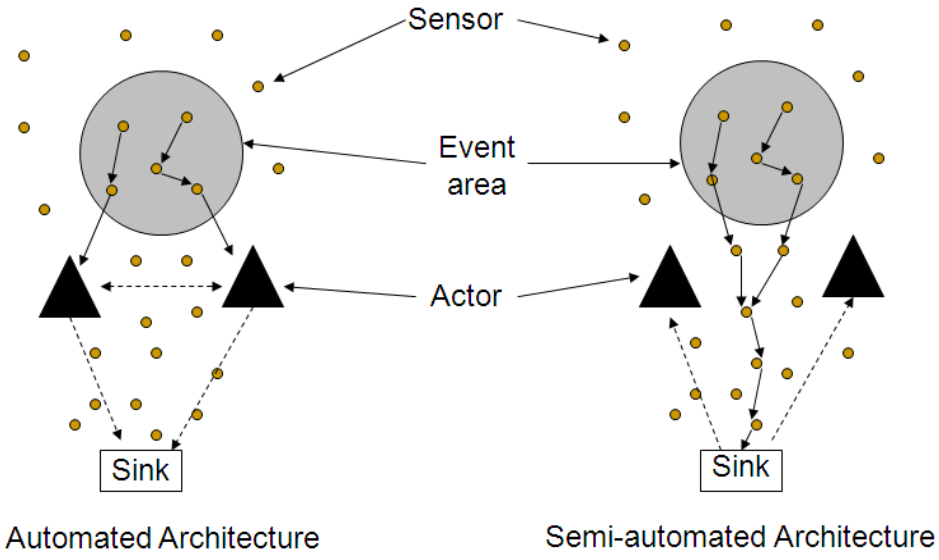

Figure 8: Architecture des réseaux de capteurs et d'actionneurs

4.1. Architecture semi-automatique :

L'architecture semi-automatique est basée sur un contrôleur central appelé le nœud puits (Sink[1]). En d'autres termes, tout doit passer par la source, collecter les données des différents micro-capteurs et communiquer les commandes à exécuter aux actionneurs. Ce rôle de coordinateur nécessite beaucoup de ressources (puissance de calcul, énergie…). C'est pourquoi, le nœud puits est généralement fixe et richement équipée.

Cette architecture est similaire à celle adoptée pour les réseaux de capteurs (WSN), elle ne prend pas en compte les actionneurs [AkK04]. Ainsi, il n'y a pas besoin de développer de nouveaux algorithmes et protocoles de communications et de coordinations.

Du fait qu'elle ne prend pas en compte l'hétérogénéité du réseau, cette architecture peut engendrer une charge de communication importante sur les micro-capteurs et peut causer ainsi la déconnexion du réseau Figure 9.

[1] Sink ou Puits : ou encore Station de base (SB) c'est un nœud du réseau généralement fixe avec plus de ressources que les autres nœuds du réseau (Sensor/Actor), il joue des rôles différents suivant l'architecture adoptée.

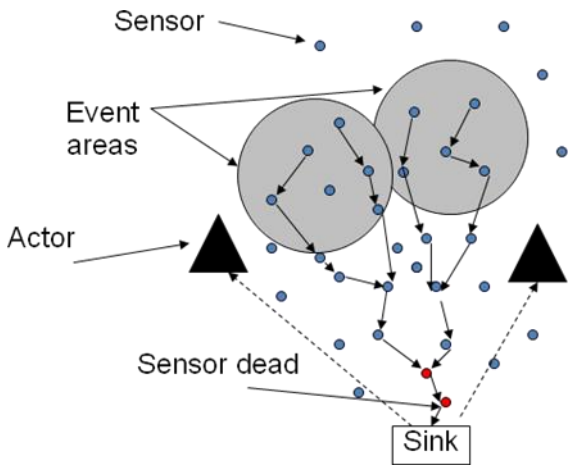

Figure 9: Architecture Semi-automatique (Nœuds Défaillons).

4.2. Architecture automatique :

Dans ce type d'architecture les micro-capteurs se chargent de transmettre les données aux actionneurs. Ces derniers disposent d'une durée de vie et d'une portée de transmission plus grande. Ils peuvent, suivant le domaine d'application, transmettre efficacement ces données au nœud puits, ou encore se coordonner entre eux pour exécuter une action bien définie.

Cette approche est plus efficace essentiellement sur deux (02) points importants :

- Délai de latence court : Les données captées sont acheminées du micro-capteur à l'actionneur instantanément Figure 8. Ainsi, la latence est minimisée dans le cas d'une architecture automatique.

- Durée de vie du réseau plus grande : L'architecture semi-automatique peut causer la déconnexion du réseau Figure 9. Due au fait que les nœuds à un saut de la source sont souvent trop sollicités. Dans une architecture automatique, différents évènements sont rapportés à différents actionneurs. De plus, les actionneurs peuvent être mobiles, ce qui diminue la charge de communication pour les micro-capteurs à un saut d'un actionneur et assure une durée de vie plus longue au réseau.

5. Les domaines d'applications

Les réseaux de capteurs et d'actionneurs sont des réseaux hétérogènes, qui peuvent être considérés comme l'union entre un réseau de capteur simple WSNs et un réseau ad-hoc avec un nombre limité de nœuds actionneurs. Ces derniers, peuvent être mobiles, et ont pour charge de récolter les informations transmises par les capteurs, les traiter et lancer les actions appropriées pour répondre à des phénomènes physiques suivant le domaine d'application.

5.1. Applications militaires :

Les applications militaires sont très liées au concept des WSNs. Le domaine d'intérêt est très étendu telle la collecte des informations (l'espionnage), la poursuite de l'ennemi, la surveillance des champs de bataille et la classification des cibles. Certaines de ces applications peuvent s'automatiser avec l'utilisation d'actionneurs. Par exemple, la détection des mines qui peut être effectuée par des actionneurs tels "Robotic Mule" Figure 5.

5.1.1. Contrôle environnemental d'intérieur :

Un bon nombre de micro-capteurs peut trouver sa place dans le contrôle de l'environnement d'intérieur tels les capteurs de températures, de lumières, de pollution et d'humidité. Ces derniers peuvent servir à informer des actionneurs comme l'extincteur d'incendie ou le contrôleur de la climatisation. L'application des WSANs peut être très bénéfique surtout si on sait qu'une grande partie de l'énergie est gaspillée, faute d'éclairer ou de climatiser un endroit vide ou avec une fenêtre ouverte.

Autre les applications de sécurité des WSANs qui peuvent détecter une fuite de gaz ou une intrusion, l'actionneur peut couper l'arrivée du gaz et ventiler la pièce, ou dans l'autre cas, fermer les sorties et répandre un gaz tranquillisant.

5.1.2. Applications à l'agriculture :

Le déploiement d'un WSANs dans l'agriculture peut permettre d'augmenter la production et améliorer la qualité des produits. La capture des informations sur l'humidité, la pluie, la température de l'air et du sol, qui peuvent être transmises à des

actionneurs qui réagissent pour assurer des conditions idéales pour le type de produit en exploitation.

Les WSANs peuvent aussi se révéler très économique en gérant les systèmes d'irrigation. Surtout que dans le domaine de l'agriculture l'eau est une ressource très importante.

5.2. Support logistique :

Le contrôle de l'inventaire est un problème majeur pour les grandes compagnies où la gestion logistique (pièces des équipements et des machines, différents types de stocks et de produits) peut être très difficile. Le problème est la vaste distribution de ces grandes compagnies à travers le monde combiné à une grande hétérogénéité des produits (produits fragiles, produits chimiques qui peuvent entrer en réaction).

Une idée prometteuse dans la gestion active des ressources logistiques est l'utilisation des balises RFID[2] combinées avec des WSANs.

Les grandes structures, telles les pipelines, les ponts et les grandes machines sont aussi au centre des recherches pour la protection et la maintenance de ces structures généralement très coûteuses. Il est plus facile et moins coûteux de renforcer une structure qui a subi des dommages que de la reconstruire, ceci dit il faut avoir des informations très précises sur les dommages subis, d'où le rôle des WSANs.

5.3. Applications médicales :

La médecine et le système de santé peuvent aussi profiter de l'application des WSANs. Les capteurs peuvent capter des paramètres, tels le rythme cardiaque et le taux de sucre dans le sang, et transmettre ces informations à des actionneurs qui interviendront avant que la situation ne devienne critique.

[2] RFID : (Radio Fréquence Identifiant) utilisé pour identifier des objets.

5.4. Applications à la robotique :

Beaucoup d'applications de robotique utilisent les WSANs, qui représentent un domaine d'application naturel où les robots eux-mêmes peuvent être des actionneurs. Un exemple, *Robomote* qui est un robot miniature développé par USC (centre de robotiques et les systèmes embarqués), peut se servir d'informations venantes de capteurs distribués pour parcourir son chemin. Cette application pourra promouvoir les recherches dans WSANs avec les robots [ALM05]. Citons comme exemple les SKITs qui représentent un ensemble de mini robots capables de communiquer en réseau et de coordonner des actions Figure 10. Ces applications pourront promouvoir les recherches dans les WSANs avec des robots comme actionneurs.

Figure 10: Sub kilogram intelligent télé-robots (SKITs)

6. Le Clustering dans les WSANs:

Le clustering est l'approche standard pour permettre aux WSANs de passer à l'échelle. Pour réaliser un contrôle efficace à travers le clustering, certaines propriétés doivent être vérifiées :

- Le clustering doit être rapide et local : la formation d'un cluster ou sa disparition ne doit pas affecter tous le réseau.
- L'algorithme de clustering doit produire des clusters de tailles équivalentes.
- Le chevauchement des clusters doit être minimal

Ces clusters peuvent être formés suivant différentes manières [MPGA07] :

- **Statiques au déploiement des capteurs :** c'est-à-dire que les capteurs sont configurés avant leur déploiement de manière à connaitre le nœud actionneur auquel ils doivent transmettre la donnée captée. Cette technique facile à mettre en œuvre, ne permet pas de prendre en charge la mobilité des nœuds.

- **Dynamique au déclanchement de l'évènement** suivant l'un de ces critères [MPGA07] :

  1) Minimiser le temps de transmission des données, ce qui diminue le temps de latence, crucial dans les WSANs.

  2) La transmission des données entre le capteur et l'actionneur se fait sur le chemin d'énergie minimum. Cela permet un gain considérable d'énergie.

  3) La zone de réaction des actionneurs peut couvrir toute la région de l'évènement pour assurer une bonne efficacité de réaction.

Le clustering représente une très bonne alternative à la contrainte de scalabilitée (ou le passage à l'échelle), dans le cas de réseau à très grand nombre de capteur et d'actionneurs.

7. La sécurité dans les WSANs :

La sécurité est importante dans beaucoup de domaines d'application des WSANs, civils ou militaires, telle la détection de catastrophes (tremblements de terre, incendies, etc.), la prévention des attaques terroristes, le contrôle de la fabrication industrielle et autres.

L'étude de la sécurité dans les WSANs doit considérer les défis suivants :

1) Prendre en considération la coordination entre capteurs et actionneurs et l'hétérogénéité entre les nœuds.
2) La simplicité du protocole car les capteurs ont une mémoire et des capacités de calcul limités [ASSC02].
3) La scalabilitée de l'algorithme : les capteurs peuvent être des milliers dans un WSAN typique.
4) Minimiser la consommation d'énergie pour rallonger la durée de vie des capteurs. Par exemple, l'énergie nécessaire pour l'envoi d'un bit de données est la même pour exécuter plus de 1100 instructions en local [AkK04]. Un protocole sécurisé dans les WSANs doit minimiser la charge de communication.

Beaucoup de schémas de sécurité actuels sont basés sur la pré-distribution des clés [DoP03] [DDHC04]. Avant de déployer les capteurs, un sous ensemble de clés est affecté à chaque capteur, pour rendre possible le partage d'une paire de clés. Cependant, cette

pré-distribution des clés ne peut satisfaire une bonne sécurité, car un attaquant peut s'emparer d'un capteur ou d'un actionneur et lire sa liste de clés permanente. Il est alors nécessaire de changer ces clés d'un temps à autre [HSS07].

La régénération des clés, ou *Re-keying,* est importante en termes d'adaptation aux changements de la topologie de réseau due à la déconnexion des nœuds, à l'ajout d'autres nœuds ou à l'interruption de la transmission sans fil.

Dans le cas d'ajout d'un nœud au réseau, il est important de distinguer entre un nœud légitime et l'infiltration du réseau par un nœud mal veillant. De plus, le schéma de sécurité doit supporter les erreurs dues à la connexion sans fil. Ces considérations couplées aux contraintes sur les ressources (Batterie, mémoire, etc.…) rendent l'élaboration d'un schéma de sécurité pour les WSANs très difficile.

8. La pile de protocole dans les WSANs :

A ce jour, il n'existe pas de protocole standard pour les WSANs. La meilleure approche est celle présentée dans [AkK04], qui propose un modèle de protocole basé sur trois (03) plans : un plan de communication, un plan de coordination et un plan de management Figure 11 [AkK04].

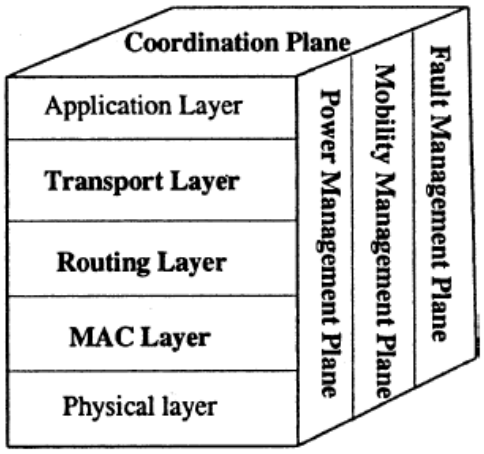

Figure 11 : Modèle de protocole dans les WSANs

8.1. Plan de management :

Les fonctions réalisées par ce plan peuvent être classées sur trois (03) axes :

8.1.1. Gestionnaire d'énergie :

Il doit définir comment un nœud utilise son énergie. La gestion de l'énergie diffère d'un nœud capteur à un nœud actionneur, mais aussi d'un nœud routeur à un nœud décideur et d'un nœud capteur fixe à un nœud actionneur mobile. Par exemple, si le niveau d'énergie est bas, le gestionnaire d'énergie peut informer le plan de coordination que ce nœud ne peut pas agir dans l'environnement, router les données ou capter les évènements.

8.1.2. Gestionnaire d'erreurs :

Il doit garantir l'intégrité du nœud, il détecte et résout les problèmes. Si un nœud capte des données erronées, le gestionnaire d'erreurs doit les détecter et informer le plan de coordination. Si le nœud est victime d'un disfonctionnement dans la transmission ou la réception des données, c'est le gestionnaire d'erreurs qui est chargé de le détecter et doit en informer le plan de coordination.

8.1.3. Gestionnaire de mobilité :

Il doit détecter et enregistrer les mouvements du nœud de manière à toujours maintenir la connectivité du réseau, ce qui n'est pas facile surtout pour les nœuds capteurs qui sont déployés en nombre important et aux faits que la gestion de la mobilité nécessite un nombre supplémentaire de messages échangés entre les capteurs et les actionneurs qui augmentes la consommation d'énergie.

8.2. Plan de Coordination :

Le plan de coordination se charge de prendre les décisions concernant des données qu'il reçoit du plan de management et du plan de coordination. Il devra implémenter tous les mécanismes et algorithmes qui permettent la coordination pour les capteurs et les actionneurs. Le plan de coordination peut se baser sur les données du plan de management et du plan de communication, pour prendre des décisions, qui peuvent être, soit une commande d'exécution de tâches, d'agrégation de données ou la sélection du nœud actionneur vers qui les données vont être transmises.

8.3. Plan de communication :

Le plan de communication reçoit les commandes du plan de coordination et les utilise pour assurer une communication efficace entre les nœuds du réseau. Il doit s'occuper de construire les canaux physiques, d'accéder aux nœuds à travers le medium (MAC), la sélection des chemins de routage des données et la transportation des paquets d'un nœud à un autre.

Dans ce qui va suivre, nous allons essayer de décrire les besoins et les défis des couches de transport, MAC et routage, mais aussi une approche d'interconnexion entre ces couches, pour les deux cas de coordination capteur/actionneur et actionneur/actionneur.

8.3.1. Couche de transport :

En plus d'assurer la bonne connectivité, le nouveau protocole doit supporter le besoin de communication temps réel qui peut être critique dans les WSANs.

Certains protocoles ont été développés pour assurer la connectivité et la communication à temps réel pour les WSANs. Dans [JGK07], des algorithmes ont été proposés en se basant sur les informations locales des nœuds, pour assurer la connectivité du réseau tout en minimisant l'énergie due aux échanges de messages.

8.3.2. Couche de routage :

Un certain nombre de protocoles de routage ont été développés pour les WSNs qui prévoient une communication à temps réel et contrôlent la congestion. Cependant aucun d'entre eux ne considère la présence des actionneurs.

Dans les WSANs, quand un nœud détecte un événement, il ne connaît pas forcément le nœud vers lequel il fera transiter les données. Ceci est dû principalement à la concurrence des nœuds dans le voisinage. De ce fait, la sélection de chemin est un défi major pour les capteurs. La donnée doit être routée suivant un chemin d'énergie minimum. Lorsqu'une donnée est envoyée en direction d'un actionneur à travers des nœuds capteurs, elle peut être agrégée ou diffusée dans le but de garantir une efficacité énergétique.

En plus de déterminer le meilleur chemin, un protocole de routage doit supporter une communication à temps réel, en considérant les intervalles de temps de validité de la donnée. Cependant, le protocole de routage peut instaurer un système de priorité pour acheminer la donnée avec un court délai, pour permettre à l'actionneur de réagir à temps.

8.3.3. Couche de contrôle d'accès au média (MAC) :

Le protocole MAC est nécessaire dans le but d'une transmission effective d'information sur un événement d'un grand nombre de capteurs vers des actionneurs. De plus, dans certaines applications telles la robotique, les actionneurs peuvent être mobiles, et peuvent quitter une région de capteurs et entrer dans une autre région de capteurs ou être totalement déconnectés du réseau. En conséquent, une autre fonction du protocole MAC est nécessaire dans les WSANs qui est : maintenir la connexion entre les capteurs et les actionneurs mobiles. De plus, comme vu précédemment : la détection, le traitement et la délivrance d'information doivent se faire au moment opportun.

8.3.4. Cross-layering ou inter connexion des couches réseaux:

Les protocoles élaborés pour les WSNs et les WSANs sont largement basés sur la structure en couche. Cependant, cette approche s'avère non optimale et inflexible, ce qui résulte sur des performances assez pauvres pour les WSANs, dues aux contraintes de consommation d'énergie et de latence. Par conséquent, au lieu d'avoir plusieurs couches individuelles, nous pouvons avoir besoin d'un croisement de couches.

Dans les WSANs, l'une des principales causes du manque de fiabilité dans la transmission d'événement est la congestion du réseau. Dans le cas d'une grande congestion, le protocole MAC réagit localement par un back-off exponentiel [CGMT04], alors que, la couche transport réagit par une baisse des taux de transmission des données. Cependant, ces deux couches réagissent indépendamment, ce qui rend leur réaction moins efficace due à la multiplication des fonctions, ce qui engendre plus de consommation énergétique.

Dans le cross-layering, chaque protocole partage ses données avec les autres protocoles pour pallier à cette inefficacité [AkK04]. Par exemple, lors d'une grande

congestion dans le réseau, le premier à réagir est le protocole de la couche MAC. Si sa réponse n'est pas suffisante, la couche MAC informe la couche de routage de la congestion, puis, la couche de routage se réfère au plan de coordination pour dérouter le trafic à travers un autre nœud actionneur approprié. D'autre part, s'il n'existe pas d'autres alternatives de chemin, le protocole de transport peut être utilisé pour geler les transmissions pour un moment.

Un autre exemple de l'approche cross-layering dans les WSANs est l'optimisation de la taille des paquets transmis des capteurs vers les actionneurs.

9. Conclusion :

La nécessité des WSANs s'est fait sentir par ses divers domaines d'application, qu'ils soient civils ou militaires, dans un environnement d'intérieur, ou pour explorer des conditions extrêmes. Elle s'est d'abord fait sentir par son enjeu économique et par les qualités de service qu'elle permet d'obtenir en précision et en temps de réponse.

Néanmoins, cette classe de réseaux émergente soulève un certain nombre de défis, telle une coordination entre des entités hétérogènes, pour aboutir à un routage à temps réel avec une consommation énergétique minimale, afin d'assurer une durée de vie plus longue au réseau. Rajouter à cela un aspect critique dans ce type de réseaux, qui est la tolérance aux défaillances. Une défaillance dans le système de détection d'incendies et d'évacuation peut engendrer des pertes humaines.

Dans ce qui va suivre, nous allons définir une classification des défaillances dans WSAN. Nous présenterons les différentes techniques utilisées pour la prise en charge de ces défaillances. Ceci dit, la problématique de la FT aux actionneurs dans les WSANs reste bien particulière. Nous allons aussi détailler les travaux qui ont été réalisés dans ce domaine.

Chapitre II : La tolérance aux défaillances dans les WSANs

1. Introduction :

La tolérance aux défaillances est la capacité du système de délivrer un certain niveau de fonctionnalité en présence de fautes (défaillances ou pannes de ses composants) [Dem04]. Elle est devenue vitale pour les réseaux de capteurs et d'actionneurs, vu leur connexion intime au monde réel par l'utilisation des équipements capteurs et actionneurs pour surveiller et réagir dans l'entourage physique.

Les WSANs soulèvent de nouveaux challenges pour la tolérance aux défaillances. Ces réseaux sont sujets aux fautes à cause de la nature partagée du médium sans fil, en plus, les pannes des nœuds à cause d'un défaut de fabrication ou l'épuisement de l'énergie sont courantes. Mais, ce qui rend la tolérance aux défaillances encore plus critiques pour ce type de réseaux, c'est la réaction dans l'environnement physique qui effectue souvent des taches mécaniques très sensible aux pannes. Et le fait que le service offert par le réseau ne doit pas s'arrêter (tel que un système de détection d'incendie et d'évacuation) un tel système doit être toujours fonctionnel quel que soit le type de panne.

Dans ce qui va suivre nous allons présenter une classification des défaillances dans les WSANs suivant plusieurs critères, ainsi qu'une procédure générique qui permet de définir les étapes nécessaires pour aboutir à une tolérance aux défaillances efficace pour les WSANs. Par la suite, Nous présenterons les différentes techniques utilisées pour la détection et la prise en charge des défaillances dans les WSANs et plus précisément la tolérance aux défaillances des actionneurs.

2. Classification des défaillances

Les défaillances dans les réseaux de capteurs peuvent être classifiées selon plusieurs critères, tels que le type de défaillance, la fréquence d'occurrence et son impact sur le système [KPS04]. Notons que cette classification est partagée par les réseaux ad-hoc, les réseaux de capteurs et les WSANs.

2.1. Classification selon les couches :

Ce type de classification englobe quatre (04) groupes de défaillance suivant le niveau où se produit la défaillance :

- **Le niveau matériel :** Les pannes à ce niveau peuvent être causées par le disfonctionnement des composants matériels tels que : la mémoire, la batterie, le microprocesseur, l'unité de captage et l'interface radio. En ce qui concerne les actionneurs, les pannes mécaniques de l'unité d'actuation sont aussi à prendre en charge. Il existe trois principales causes de ces pannes : La première cause est que les capteurs sont destinés à l'usage commercial et ils sont sensibles au coût, ainsi ils n'utilisent pas de composants de bonne qualité. La seconde cause est le fait que les contraintes strictes en termes d'énergie rendent les capteurs moins fiables et de faible performance sur une longue durée d'utilisation. Par exemple, si la batterie d'un capteur diminue à un certain niveau, ces lectures de grandeurs physique deviennent incorrectes [TPS05]. La troisième cause est que les capteurs sont déployés dans un environnement hostile, ce qui peut affecter sévèrement le fonctionnement de ces derniers.

- **Le niveau logiciel :** Les logiciels d'un nœud capteur sont répartis en deux catégories: les *logiciels système* et les *Intergiciels*[3] de communication, routage et agrégation. Les bugs logiciels sont la source commune des erreurs dans les logiciels des réseaux de capteurs sans fil. Une manière de réaliser la tolérance aux défaillances à ce niveau est de diversifier les logiciels et que chacun soit implémenté dans une version différente.

- **Le niveau communication réseau :** Les pannes de ce niveau sont celles des liens de communication sans fil. Ce type de panne est habituellement lié aux interférences et collisions qui dégradent les performances des liens sans fil. Une manière de renforcer ces liens est d'utiliser des schémas de correction et de retransmission, mais ces méthodes causent des retards sur les opérations de communication.

- **Le niveau application :** La tolérance aux défaillances peut aussi s'adresser au niveau application. Par exemple, une application qui construit des chemins multiples de routage avec des nœuds disjoints, peut assurer une tolérance aux

[3] Intergiciel : un composant important du système logiciel qui permet de supporter les exécutions simultanées et de façon distribuées des algorithmes localisé.

défaillances dans le routage. Le système peut choisir alors d'abandonner un chemin avec un lien coupé pour un autre chemin candidat. Notons qu'une approche de tolérance aux défaillances pour une application n'est pas directement applicable pour une autre application. Chaque application demande sa propre approche de tolérance aux défaillances. Notons aussi que la tolérance aux défaillances dans la couche application traite les pannes de toute source.

2.2. Classification selon les types

Les capteurs peuvent être déployés dans un environnement hostile où les canaux de communication peuvent générer du bruit important. Trois types de défaillances ont été définis dans [KPS04], à savoir :

- **Erreur du canal (CH-ERR) :** les transmissions peuvent temporairement échouer à cause des collisions, les interférences ou les erreurs dans les circuits. Cependant, l'émetteur d'un paquet ne peut pas savoir si son paquet a été perdu durant la transmission.

- **Echec aléatoire des nœuds (RAND-FAIL) :** les nœuds en panne sont dispersés sur toute la zone d'application du réseau de capteurs. Les nœuds corrects peuvent reconnaître les nœuds en échec par des mécanismes systématiques tels que les messages périodique heartbeat.

- **Echec régulier des nœuds (PATN-FAIL) :** les nœuds en panne peuvent suivre un schéma d'échec (pattern failure). Par exemple, pour une application de détection d'incendie, une région de nœuds qui sera en échec à cause de l'incendie.

2.3. Classifications selon la fréquence d'occurrence

- **Pannes permanentes :** Une panne permanente, une fois qu'elle est activée, elle persiste jusqu'à ce qu'elle soit découverte et prise en charge. Nous trouvons dans cette catégorie les pannes des nœuds dues à l'arrêt d'un composant interne tel que l'unité de traitement ou l'unité de captage, ou bien l'épuisement

d'énergie. Le système de communication peut aussi souffrir de ce type de pannes à cause de la défaillance des émetteurs/récepteurs radios.

- **Pannes intermittentes :** Ce type de panne est plus difficile à détecter et à prendre en charge parce qu'il ne suit aucun modèle d'échec (pattern failure). Par exemple, un obstacle qui passe aléatoirement à travers un lien sans fil de manière répétitive peut causer une coupure de lien intermittente.

- **Pannes transitoires :** par exemple, une région de congestion dans le réseau peut empêcher les nœuds dedans à répondre aux messages de signalisation de la part de leurs nœuds voisins, ce qui peut être interprété par ces nœuds comme une défaillance transitoire.

3. Procédure de tolérance aux défaillances :

Une procédure générique de tolérance aux défaillances se divise en deux parties : Une première partie d'apprentissage appelée *gestion de la tolérance aux défaillances,* qui définit les règles d'opération pour la seconde partie appelée E*xécution de la tolérance aux défaillances* Figure 12.

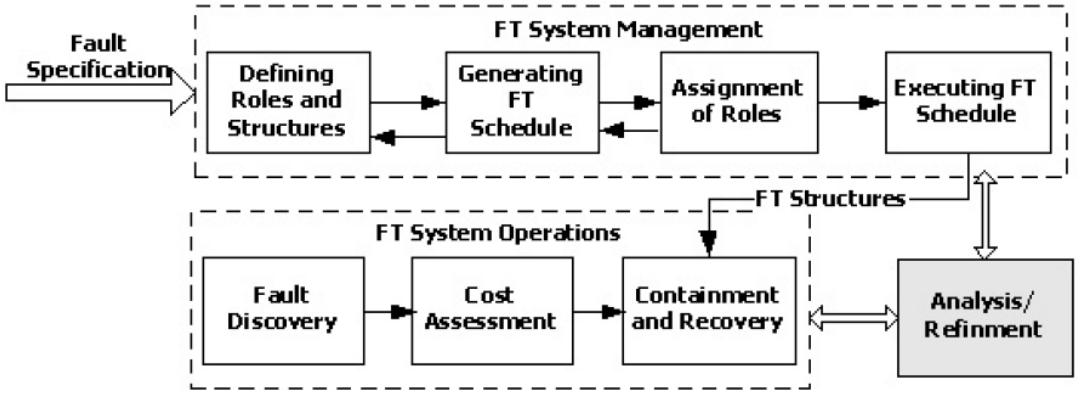

Figure 12 : Plateforme pour la tolérance aux défaillances

La gestion de la tolérance aux défaillances reçoit en entrée la spécification de la tolérance aux défaillances requise, qui inclut les informations fournies par le concepteur du système, telles que la nature des pannes et leurs fréquences. Ceci inclut aussi les ressources qui doivent être tolérantes aux défaillances. Par exemple, une règle de la

forme « les capteurs d'une même région d'événement délivre les même données captées » peut être définie comme un fonctionnement normal du système.

Une procédure de tolérance aux défaillances doit implémenter les étapes suivantes :

1ère partie :

- **Définition des rôles et de la structure :** un rôle définit une ou plusieurs tâches qui doivent être exécutées par un ou plusieurs composants. La réplication des données, la surveillance des ressources et la découverte de pannes sont des exemples de rôles. Un exemple de structure est la donnée répliquée ou la structure de clustering. La définition des rôles et des structures peut être faite à l'initialisation du système, ou peut être adaptative.

- **Génération du plan de tolérance aux défaillances :** les tâches de tolérance aux défaillances peuvent être planifiées suivant les conditions du réseau et les performances attendues. Par exemple, un plan de réplication de données est basé sur la fréquence des résultats demandés et les mesures du réseau.

- **Affectation des rôles :** c'est la correspondance entre les rôles et les composants du système. On classifie les schémas de la tolérance aux défaillances en deux types : in-network et off-network.
  - ***In-network*** affecte les rôles aux capteurs tout en étant conscient de la tâche du réseau. La tolérance aux défaillances dans ce cas consomme les ressources du réseau comme toute autre fonctionnalité du système.
  - ***Off-network*** affecte les rôles à quelques nœuds spéciaux. Ainsi, le schéma est inconscient de la mission du réseau.

Un troisième schéma peut être défini par l'hybridation des deux schémas.

2ème partie :

- **Exécution du plan de tolérance aux défaillances :** c'est l'application des tâches nécessaires pour maintenir le système tolérant aux défaillances. L'exécution peut être faite une seule fois ou bien d'une façon périodique.

  L'exécution de la tolérance aux défaillances consiste en :

  - ✓ ***Découverte de pannes :*** une panne d'un composant du système peut être détectée si le comportement du composant est différent de ce qui est spécifié.
  - ✓ ***Evaluation du coût :*** concerne l'estimation des dégâts et l'évaluation des alternatives possibles. Le but de ce module est la déduction d'un plan de reprise optimal.
  - ✓ ***Contamination et reprise après pannes*** : une fois le plan est établi, il est exécuté en éliminant les effets de la panne. Ce module utilise la structure définie.

- **Analyse et raffinement :** ce module est critique pour la tolérance aux défaillances, car il permet au réseau de s'auto-configurer en se basant sur le feedback des opérations de tolérance aux défaillances. Par exemple, en se basant sur la fréquence de pannes, une régénération du plan peut être déclenchée. L'affectation de rôles peut être aussi réappliquée. Les schémas de tolérance aux défaillances peuvent être classés en deux groupes : statique et dynamique, selon la façon d'analyse et de raffinement.

Une bonne tolérance aux défaillances doit implémenter tous les modules précédant et doit être la plus dynamique possible, pour s'adapter à la mobilité des nœuds, surtout les actionneurs qui doivent se déplacer pour couvrir une région plus grande.

4. Prise en charge des défaillances

Pour faire face aux défaillances dans les WSANs, le système doit suivre deux étapes. La première étape est la détection de la défaillance. Elle consiste à détecter qu'une fonctionnalité du système n'est pas conforme et qu'un fonctionnement correct est possible dans le futur proche. La deuxième étape est la reprise après panne. Elle

permet au système de reprendre son fonctionnement normal après chaque panne détectée.

Il existe deux méthodes de détection de pannes des nœuds capteurs : l'auto diagnostique et le diagnostic coopératif. Quelques pannes liées au nœud lui-même, telles que la détection du niveau de batterie ou la détection de l'échec d'un lien, sont auto diagnostiquées par le nœud lui-même.

Cependant, une grande partie des défaillances dans les réseaux de capteurs sans fil sont diagnostiquées de manière coopérative. Par exemple, une méthode de détection des défaillances [KrI04] suppose que les lectures de tous les capteurs dans une région donnée sont fortement corrélées. Ainsi, si la lecture d'un nœud n'est pas conforme, la méthode trouvera avec une grande probabilité le nœud défaillant.

Les méthodes les plus utilisées pour la reprise après panne sont la *réplication* et la *redondance*. Par exemple, dans la majorité des cas, un WSAN est utilisé pour surveiller une région donnée et transmettre les données à l'actionneur. Lorsque certains nœuds tombent en panne, l'actionneur peut recevoir suffisamment de données à partir des nœuds redondants pour lui permettre de réagir dans l'environnement suivant les données captées sur l'événement en question.

Le routage multi-chemin dans les réseaux de capteurs est un autre exemple. La construction d'un ensemble de chemins candidats fournit une fiabilité importante pour le routage, et tolère des pannes de liens et de nœuds. Ceci exige une k-connectivité de nœuds pour tolérer k-1 pannes de nœuds.

4.1. Placement des nœuds relais :

L'architecture à deux tiers emploie des nœuds relais puissants pour collecter les informations des autres nœuds capteurs du réseau, les traiter et les transmettre au nœud puits. Donc, le réseau est organisé en clusters et les nœuds relais sont considérés comme clusterheads. Une panne de communication (émission/réception) d'un nœud relais exige des capteurs d'un cluster de changer de clusterhead. Si les pannes se produisent entre les clusterheads, ces derniers doivent avoir un chemin alternatif pour communiquer. En général, pour prendre en charge les pannes de communication, il doit y exister au moins deux chemins disjoints (en termes de nœuds) entre chaque pair de

nœuds du réseau. Le but recherché est de minimiser le nombre de nœuds relais pour tolérer un certain degré de défaillances.

Il existe plusieurs variantes de définition du problème de placement des nœuds relais. En général, on considère un ensemble de nœuds capteurs répartis aléatoirement dans une région. Certains nœuds relais sont nécessaires pour expédier les données captées au nœud puits tel que chaque capteur doit être couvert par au moins un nœud relais. Le but recherché est de minimiser le nombre de nœuds relais pour rendre le réseau k-connecté (généralement 2-connecté).

4.2. Contrôle de topologie :

Le placement des nœuds relais peut fournir une certaine tolérance aux défaillances dans les WSANs, mais ces nœuds relais ne peuvent ne pas jouer leur rôle correctement à cause de la mobilité. Pour ceci, le contrôle de topologie est nécessaire pour maintenir la propriété de tolérance aux défaillances dans les réseaux de capteurs.

Une approche pour réaliser le contrôle de topologie est d'utiliser le concept des CDS (Connected Dominating Set) [ChS05]. Le CDS est considéré comme l'épine dorsale du réseau. Chaque nœud dans le CDS ajoute les nœuds voisins à l'épine dorsale de telle sorte que le réseau soit k-connecté (pour un k donné). Des métriques sont utilisées pour sélectionner les nœuds qui participent au CDS, telles que : l'énergie, le degré de connexion, et des métriques hybrides.

Dans [CYW07], les auteurs ont étudié le contrôle de topologie avec tolérance aux défaillances dans les réseaux de capteurs sans fil. Le réseau est constitué de deux types de nœuds : un grand nombre de nœuds capteurs, et un nombre réduit de super nœuds. Le problème est d'ajuster la puissance de transmission de chaque nœud de telle sorte d'avoir k chemins disjoints (en termes de nœuds) entre chaque nœud capteur et l'ensemble des supers nœuds.

Le but recherché est de minimiser l'énergie totale consommée par les nœuds capteurs. Une des solutions proposées est un algorithme k-approximatif en deux étapes. La première étape consiste à réduire le graphe en un graphe direct avec les supers nœuds dans la racine. La deuxième étape est une solution du problème min-Wright-k-Outconnectivity, qui est adaptée pour calculer le rayon de transmission minimal pour chaque nœud capteur. Ces deux étapes sont détaillées dans [LNS07].

5. Tolérance aux défaillances des actionneurs :

La tolérance aux défaillances dans les réseaux de capteurs n'est pas directement applicable dans le cas WSANs dû principalement à l'hétérogénéité des nœuds et l'existence des actionneurs, qui doivent aussi être tolérants aux défaillances. De plus, ces actionneurs sont plus sensibles aux pannes vu les taches qu'ils doivent effectuer.

La tolérance aux défaillances est encore plus difficile à assurer dans un modèle SAMS (Single Actor Multi Sensors) Figure 13, où l'actionneur en question devient un point de convergence critique. En cas de panne de l'interface radio de l'actionneur, les autres actionneurs ne pourront jamais faire la reprise après panne par manque de données sur l'évènement ou encore parce qu'ils ne peuvent savoir si l'actionneur responsable de la région d'évènement a déjà réagi à l'évènement ou pas. Dans le cas où l'actionneur sélectionné subit une panne sur l'unité d'actuation, il ne pourra pas réagir à l'évènement mais il peut toujours lancer une négociation avec les autres actionneurs de la région pour réagir à l'évènement, ce qui augmente le temps de latence de réaction.

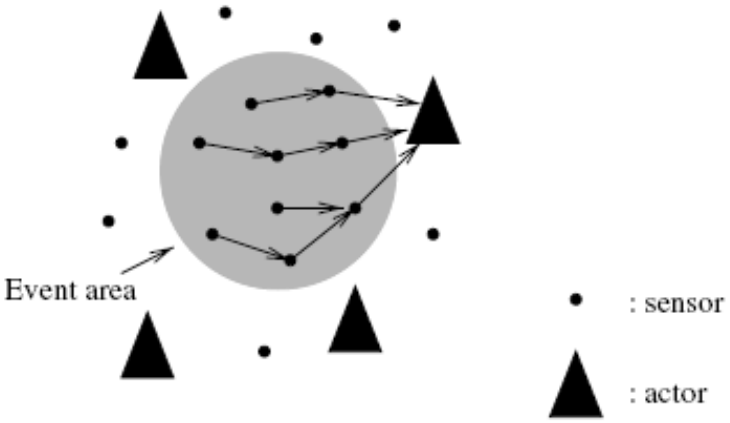

Figure 13 : Single Actor

Il semble plus évident que l'utilisation d'un modèle multi actionneur multi sensors MAMS est plus efficace pour assurer une tolérance aux défaillances, plusieurs actionneurs peuvent recevoir les informations des capteurs, où chaque nœud capteur peut indépendamment décider à quel actionneur il envoie les informations captées Figure 14. L'utilisation d'un model MAMS pour assurer une FT des actionneurs va être détaillé dans ce qui va suivre.

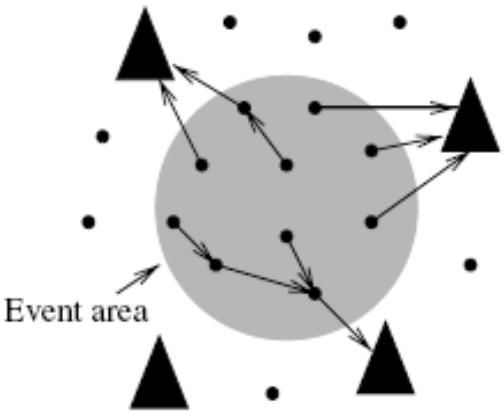

Figure 14 : Multi Actors

Dans [OHE07], un travail a été réalisé pour assurer la tolérance aux défaillances des actionneurs dans un WSAN. Les auteurs on utilisés un model MAMS avec une coordination semi-passive (SPC) qui ce vois être la plus efficace parmi les quatre type de coordination qu'ils ont identifié [OHE07]. Le principe de fonctionnement est simple, il se base principalement sur l'élection d'un actionneur primaire unique qui prend la décision de la tache à exécuter suivant les données captées. Les autres actionneurs backup reçoivent les données captées et les transmettent directement à l'actionneur primaire. En cas de défaillance du primaire, un algorithme de consensus est exécuté pour choisir un autre primaire parmi les backups. Cette technique permet d'assurer la tolérance aux défaillances des actionneurs, mais avec une certaine latence (temps de choisir un remplaçant à l'actionneur primaire). De plus, en cas de panne de l'interface radio de l'actionneur primaire, les backups ne peuvent pas détecter la panne.

A l'heure actuelle peu de travaux traitent la problématique de la tolérance aux défaillances dans les WSANs. Comme nous l'avons vu plus haut dans ce Chapitre, les techniques proposées pour la tolérance aux défaillances dans les WSNs ne s'appliquent pas aux WSANs. De plus, les défaillances des actionneurs ne sont pas prises en charge, malgré l'importance stratégique de ces derniers pour la réaction dans l'environnement physique.

6. Conclusion

La tolérance aux défaillances soulève beaucoup de chalenge dans les WSANs, en particulier, la tolérance pour les actionneurs déployés avec un nombre limité et devant parfois être mobiles pour couvrir une région plus grande. Ces actionneurs, en plus des pannes traditionnelles de batterie et d'interface radio, sont sujets à des pannes mécaniques sur l'unité d'actuation.

Comme nous avons pu le voir dans ce Chapitre, les techniques de redondance et de réplication ne sont pas adaptées pour tolérer les pannes des actionneurs : la première exige un grand nombre d'actionneurs, et la seconde augmente la consommation d'énergie des capteurs qui font transiter la donnée dupliquée entre les différents actionneurs.

Des techniques nouvelles doivent être proposées pour tolérer les pannes temporaires ou définitives des actionneurs, tout en minimisant, le nombre d'actionneurs à déployer, le temps de latence d'exécutions des actions et l'énergie consommée par les capteurs, qui eux aussi doivent être tolérant aux défaillances.

Durant notre recherche nous avons constaté qu'il existe très peu de travaux sur la FT des actionneurs dans les WSANs malgré l'importance stratégique de ce domaine bien particulier. Dans ce qu'il va suivre nous allons proposer une nouvelle technique de tolérance aux défaillances des actionneurs basée sur le clustering. Cette technique implémente aussi un protocole de routage avec une QoS qui nous permettra d'optimiser les temps d'exécutions des actions et l'énergie consommé dans le réseau.

Chapitre III : Proposition d'une nouvelle technique de clustering et d'un protocole de routage

1. Introduction :

Les WSANs trouvent leur place dans un grand nombre de domaines d'applications, et chacun impose des contraintes supplémentaires telles la communication à temps réel et la tolérance aux défaillances *Fault Tolerance* (**FT**). Un exemple est la détection et l'extinction des incendies de forêts, où le temps de latence entre le captage de la donnée et l'intervention de l'actionneur doit être minimal, pour éviter la propagation du feu, d'où le transfert des données captées en temps réel. Ainsi, la tolérance aux défaillances des actionneurs est cruciale pour réagir efficacement à l'événement capté.

Le souci d'automatisation d'un tel système, et ainsi, le rendre fonctionnel quelque soient les défaillances auxquelles il est confronté, exige une tolérance à tous les types de défaillances et sur tous les nœuds présents dans le réseau.

Peu de travaux ont été réalisés pour répondre au problème de la tolérance aux défaillances dans les WSANs. A l'heure actuelle, il existe un travail qui traite le problème épineux de la tolérance aux défaillances des Actionneurs dans les WSANs. Dans [OHE07], les auteurs proposent un protocole de coordination Semi-passive (SPC), pour une tolérance aux défaillances des actionneurs dans un modèle de communication Multi Actionneurs Multi Capteurs (*Multi Actor Multi Sensor)* **MAMS**. D'autres travaux sur la **FT** ont été réalisés dans les WSN, mais tous ces travaux ne prennent pas en charge les pannes des actionneurs, ni l'hétérogénéité des nœuds.

Dans ce qui suit, nous allons proposer des méthodes pour assurer une tolérance aux défaillances des actionneurs, mais aussi un nouveau algorithme de clustering qui permet de devisé le réseau en plusieurs cluster autoréparable. Un protocole de routage qui intègre une politique de QoS est aussi proposée dans le but de permettre de minimiser les temps de latences d'exécutions des actions.

2. Schéma de déploiement :

Sur le terrain, un réseau de capteurs et d'actionneurs peut se décliner en deux schémas de déploiement. Le premier est le modèle standard avec deux types de nœuds : les nœuds capteurs et les nœuds actionneurs, qui sont chargés de traiter les données captées et réagir dans l'environnement sans l'intervention d'un équipement tiers.

Le deuxième schéma dispose de trois types de nœuds. Nous retrouvons toujours les nœuds capteurs chargés de capter les évènements et les transmettre aux actionneurs. Toutefois, la différence est justement dans l'actionneur qui se voit diviser en deux nœuds distincts et autonomes. Le premier nœud est appelé toujours actionneur ou *Actuator*, qui aura pour rôle unique la prise de décision concernant les données captées. Une fois la décision prise par l'actionneur, la tache à exécuter sera transmise à un autre nœud du réseau appelé l'équipement d'actuation ou *Actuation device,* qui sera chargé de l'exécution de la tache.

2.1. Schéma de déploiement Capteurs-Actionneurs:

Le modèle Capteurs-Actionneurs (*Sensor-Actor)* **SA** est le modèle le plus simple et le plus utilisé. Quand nous parlons de WSANs, nous visualisons directement et plus facilement un ensemble de nœuds capteurs interconnectés avec un autre type de nœuds nommés actionneurs. Ces derniers, comme défini dans le 1er Chapitre, en plus des unités de traitement et de transmission, disposent d'une unité de décision et une unité d'actuation. Cette dernière est chargée de convertir le signal numérique en un signal analogique et le transmettre à l'équipement d'actuation [AkK04].

Dans un schéma de déploiement **SA**, l'unité d'actuation est une partie intégrante de l'actionneur. De ce fait, le déploiement d'actionneurs coûte cher, ce qui exclut la méthode de FT par redondance. De plus, ce type d'actionneur, est sujet à deux types de pannes : les pannes liées à l'actionneur comme les pannes de transmission, CPU, mémoire, et les pannes liées à l'équipement d'actuation telles que les pannes mécaniques. Dans les deux cas, l'actionneur devient non-opérationnel et ne peut pas réagir dans l'environnement.

L'intérêt de séparer la décision de la réaction dans l'environnement parait juste et appuyée quand il s'agit d'assurer une tolérance aux défaillances des nœuds actionneurs dans un WSAN.

2.2. Schéma de déploiement Capteurs-Actionneurs-Equipements d'Actuation:

Dans ce modèle, un réseau de capteurs et d'actionneurs ***W*** est constitué de trois (3) types de nœuds : les nœuds capteurs, les actionneurs et les équipements d'actuations, sur lesquels on exécute les taches. Les équipements d'actuations reçoivent les signaux analogiques transmis par l'unité d'actuation et exécutent l'action appropriée[4]. Ces trois types de nœuds sont interconnectés à travers un réseau sans fil [OHM07, ASC02, MPG05]. Un nœud capteur capte des valeurs du monde physique et les envoie à l'actionneur. Ce dernier décide quelles actions exécuter pour les valeurs captées, et les exécute sur les équipements d'actuations. Une différence de taille entre les deux modèles est que dans le modèle Capteurs-Actionneurs-Equipements d'Actuation (*Sensors-Actors-Actuation device)* **SAA**, un actionneur peut contrôler plusieurs *Equipements d'Actuation* et un *Equipements d'Actuation* peut être contrôlé par plusieurs Actionneurs ou *Actuators* [OWI06] Figure 15. Ceci représente une donnée importante pour la **FT,** mais qui pose le problème de consensus entre les différents actionneurs de la région.

Posons ***S*** l'ensemble de nœuds capteurs, soit ***A*** l'ensemble des Actionneurs, et ***O*** l'ensemble des Equipements d'Actuation.

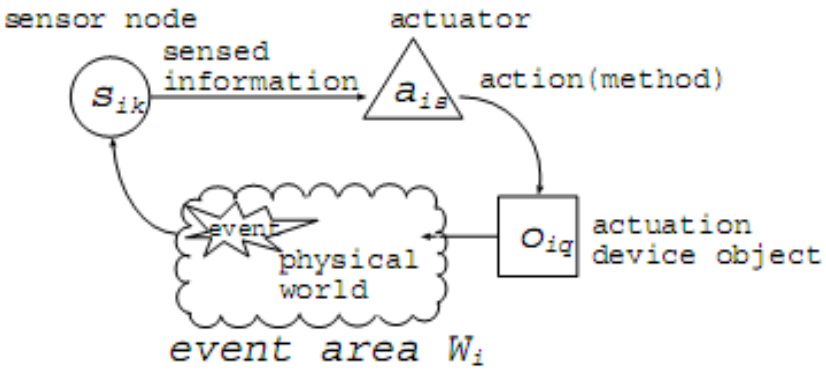

Figure 15 : Interconnexion entre les capteurs, les actionneurs et les équipements d'actuations

[4] Chapitre 1 : Architecture d'un actionneur

Un WSAN est partitionné en plusieurs zones d'événements ou *event area* $\boldsymbol{W_1 \ldots W_m}$ avec m >1. Une zone d'événements ***Wi*** est une unité géographique qui est composée d'au moins un capteur, et peut être couverte par au moins un actionneur et une unité d'actuation. ***Wi*** est représentée ainsi ***Wi = {Si, Ai, Oi},*** où les capteurs de ***Si*** captent les événements dans la zone d'événements ***Wi***, les valeurs captées sont envoyées vers les actionneurs ***Ai*** et ces derniers exécutent des actions sur les équipements d'actuation ***Oi***.

Les équipements d'actuations qui ont le rôle d'agir sur le monde physique, tel qu'un système de conditionnement d'air, un système d'arrosage ou une caméra, sont représentés comme des objets. Un objet est une encapsulation d'états et de méthodes pour la manipulation du monde physique. La réaction dans l'environnement physique est représentée par une exécution des méthodes sur un ou plusieurs équipements d'actuations dans l'ensemble d'équipements d'actuations ***Oi***.

L'actionneur délivre les méthodes à exécuter à l'équipement d'actuation. A ça réception, la méthode est exécutée sur l'équipement d'actuation. Par exemple, un objet qui représente un système de conditionnement d'air qui reçoit une méthode *Turn on(24),* s'allume et ventile l'air frais à une température de *24°C*.

Dans ce type de modèles, plusieurs actionneurs peuvent exécuter des méthodes dans l'objet qui représente l'équipement d'actuation. Nous considérons que même si plusieurs actionneurs envoient la même méthode, cette méthode ne sera exécutée qu'une seule fois sur l'équipement d'actuation. De plus, si un actionneur envoie des méthodes qui sont en conflit avec les méthodes envoyées par d'autres actionneurs, ces méthodes seront sérialisées sur l'équipement d'actuation. Dans [NaT94], les auteurs discutent l'exécution de méthode redondante et la sérialisation des méthodes venant de plusieurs actionneurs.

2.3. Choix du schéma de déploiement :

Il est évident que le modèle **SAA** peut être mieux adapté pour la tolérance aux défaillances des actionneurs, du fait que, la prise de décision et l'exécution de la tache se font sur deux types de nœuds différents. En cas de panne d'un équipement d'actuation dans ***Oi***, l'actionneur dans ***Ai*** peut toujours choisir un autre équipement d'actuation dans l'ensemble ***Oi***. De plus, un équipement d'actuation dans ***Oi*** peut se mouvoir pour couvrir une plus large *zone d'événement* dans laquelle il doit agir, alors qu'un Actionneur restera

conditionné par la *zone de contrôle* en termes de capteurs. Suivant le domaine d'application d'un WSAN, les couts de déploiement des actionneurs et des équipements d'actuation. Nous pouvons retrouver le cas où la *zone de contrôle d'un actionneur* est plus grande que la *zone d'événement* des équipements d'actuation ou le contraire.

Dans ce qui va suivre, nous allons présenter le modèle *Multi Actor Multi Sensor* MAMS, qui définit le comportement des capteurs par rapport aux actionneurs.

3. Schéma de déploiement et le modèle de communication:

Dans un modèle de communication un seul actionneur multi capteurs *(Single Actor Multi Sensors)* **SAMS**, il existe un seul actionneur ***A*** et un ensemble de capteurs ***Si*** dans une zone d'événements ***Wi***. Si un événement se produit dans la zone d'événements ***Wi***, les capteurs dans ***Si*** captent des valeurs de l'événement, telles que la température et l'humidité, et les envoient à un seul actionneur ***A***. Ce modèle est simple à déployer sur terrain et moins coûteux, mais il n'assure pas la tolérance aux défaillances des actionneurs. En cas de panne du système de transmission de ***A*** dans un schéma de déploiement SAA (*Sensors-Actors-Actuation devices)*, même si la décision de la méthode à exécuter sera prise par l'unité de décision de l'actionneur, elle ne pourra pas être transmise et les autres actionneurs ne sauront pas qu'un événement s'est déclenché puisqu'ils ne sont pas destinataires des messages transmis par les capteurs dans ***Si***.

Par opposition à ce modèle SAMS, un autre modèle **MAMS** plus tolérant aux pannes des actionneurs peut être défini, où dans une région d'événements ***Wi,*** chaque capteur dans ***Si*** envoie les valeurs qu'il a capté d'un événement ***E*** à un actionneur dans ***Ai,*** et chaque actionneur de ***Ai*** aura une partie de l'information sur l'événement. Une phase de coordination doit être lancée entre tous les actionneurs de ***Ai*** pour pouvoir prendre une décision sur quelle méthode exécuter et sur quel équipement d'actuation dans ***Oi*** [OWI06]**.**Figure 16.

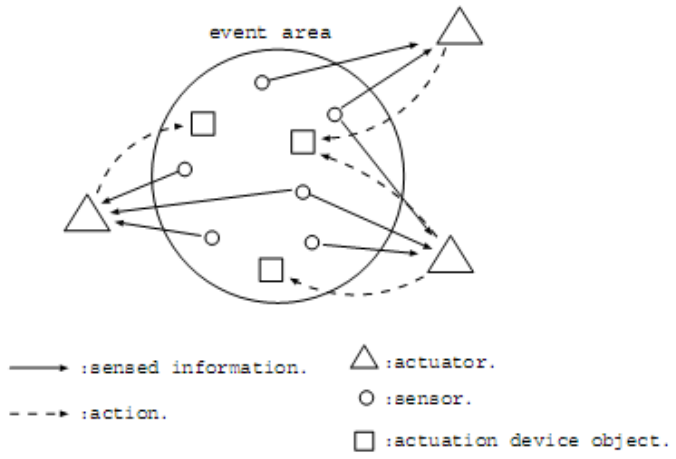

Figure 16 : Schéma de déploiement et le modèle MAMS

4. Techniques proposées pour assurer une FT des Actionneurs:

Nous avons proposé une solution de tolérance aux défaillances basée sur des *clusters autoréparables* ***CA***. Nous utiliserons la mobilité des actionneurs dans un modèle **SA**, ou plus précisément, des équipements d'actuations dans un modèle **SAA**, pour tolérer les pannes des actionneurs dans les WSANs.

4.1. Mobilité de l'action :

Dans un schéma de déploiement SA, les actionneurs peuvent être mobiles et de cette façon couvrir une plus grande zone d'événements. Cette mobilité est une mobilité de l'action[5] dans l'environnement, elle peut se réaliser sans le déplacement du nœud actionneur dans certains cas. Par exemple, une caméra qui peut zoomer sur différentes zones d'événements, ou un système d'arrosage qui peut changer de pression pour atteindre des zones d'événements plus éloignées tout en restant fixe, ou encore un actionneur qui se trouve être relié en réseau filaire à plusieurs systèmes qui peuvent être gérés à distance . Toutefois, les actionneurs qui sont naturellement mobiles, tels que les robots qui peuvent se déplacer et réagir dans des zones d'événements différentes, sont aussi pris en compte dans ce type de mobilité de l'action.

La mobilité de l'action des actionneurs apporte un grand avantage dans les WSANs, du fait qu'elle permet de minimiser le nombre d'actionneurs déployés sur le

[5] Mobilité de l'action : l'actionneur réagit dans différentes zones d'événements sans se déplacer.

terrain. En effet, nous pouvons profiter, suivant le type d'application des WSANs, de la corrélation temporelle[6] ou spatiale[7] entre les événements pour déployer un minimum d'actionneurs et utiliser la mobilité de ces derniers pour couvrir une zone d'événement plus grande. Dans ce qui va suivre, nous allons utiliser ce type de mobilité pour construire ce que nous avons appelé des C*lusters Autoréparables*.

Dans un schéma de déploiement **SAA**, la mobilité de l'action se fait naturellement sans le déplacement de l'actionneur. Si on considère que la réaction dans l'environnement est effectuée par les équipements d'actuations, un actionneur peut prendre la décision par rapport aux méthodes à exécuter en fonction de l'événement capté, et charger un ou plusieurs équipements d'actuation d'exécuter ces méthodes. Ces équipements d'actuation peuvent avoir une mobilité de l'action et réagir dans une zone d'événements plus grande.

L'avantage du schéma de déploiement **SAA** est que la transmission des données captées vers l'actionneur, que ça soit dans un modèle SAMS ou en MAMS, se fera sans coupure. Les capteurs savent toujours comment joindre les actionneurs vu qu'ils seront fixes, et les actionneurs peuvent utiliser un algorithme de *tracking* [FWW06, HaZ07] pour suivre le déplacement des équipements d'actuations et ainsi, garder une communication en temps réel.

4.2. Cluster Autoréparable :

Un autre atout des WSANs qui vient s'ajouter à la mobilité d'action des actionneurs, le clustering qui se révèle être une très bonne solution pour le passage à l'échèle dans un WSAN.

Dans un cluster, nous retrouvons habituellement trois types de nœuds qui jouent des rôles différents et complémentaires. Le premier type de nœud est le Chef de groupe ou *Cluster Head* **CH**, son rôle est de coordonné la communication *intra cluster.*

Il se trouve que ce type de nœud a des exigences particulières en termes d'énergie, de capacités de calcul et de transmission. Ces exigences sont bien remplies par

[6] Corrélation temporelle : par exemple, un type de plante est arrosé le matin par contre un autre type de plante est arrosé le soir.

[7] Corrélation spatiale : un feu de forêt se propage toujours en continu sur une surface. Tous les autres actionneurs de la région peuvent intervenir sur la zone incendiée.

les actionneurs qui sont eux même conçus pour avoir une plus grande capacité en énergie, calcul et transmission. Les actionneurs sont naturellement des CH.

Vu le nombre limité des actionneurs qui seront déployés sur le terrain, les clusters seront de taille conséquente. Un bon dimensionnement des capacités des actionneurs s'impose avant le déploiement d'un WSAN. Par exemple, suivant le domaine d'application, nous pouvons choisir de déployer un grand nombre d'actionneurs avec une petite puissance de calcul et de transmission, ou un nombre limité d'actionneurs avec une grande capacité de transmission. Dans les deux cas notre protocole s'adapte. Il s'appuie sur une variable que nous avons appelé *HELLO_TTL*, pour dimensionner la taille du cluster suivant la capacité de l'actionneur *Cluster Head.*

Le deuxième type de nœud c'est les nœuds de bordure qui assurent la communication *inter cluster*. Dans un WSAN, ce type de nœud est certainement un capteur qui se trouve à quelques sauts de l'actionneur CH. Ces nœuds capteurs, que nous appellerons *Sensor Border* (SB), ont une fonction très importante et doivent être choisis en premier lieu, suivant leur niveau d'énergie ; vu qu'ils vont faire transiter tout le trafic entre deux clusters voisins. Une bonne capacité de calcul et de transmission est un plus pour le choix d'un SB. Le nombre de SB est aussi important, plus le nombre de SB est grand plus la communication inter cluster sera tolérante aux pannes, sans oublier le partage de charge ou « *load balancing »* qui peut permettre d'augmenter la durée de vie des SBs en se partageant la tache de communication.

Le troisième type de nœud c'est l'ensemble des nœuds membres du cluster qui suivent le plan de communication établi par le **CH** et acheminent les données. Un plan de routage bien établi, qui prend en compte le partage de charge entre les capteurs à un saut de l'actionneur CH, peut permettre de minimiser l'énergie consommée par une communication en broadcast, surtout sur ces nœuds à un saut du **CH** qui seront les plus sollicités pour router les données captées vers le CH.

Le rôle de chaque nœud se trouvant dans un cluster ***Ci*** sera défini par l'actionneur chef de ce cluster le ***CHi,*** qui est aussi responsable de tous les nœuds capteurs ***Si*** présents dans ***Ci*** en termes de décision sur les événements ***Ei*** captés par ces capteurs. De plus, l'actionneur ***CHi,*** dans un schéma de déploiement SAA, sera aussi responsable de tous les équipements d'actuation ***Oi*** présents dans le cluster ***Ci***. Dans un

schéma de déploiement SA, le ***CHi*** ne sera responsable que de son équipement d'actuation interne.

Le clustering en un modèle SAMS avec l'actionneur comme CH ne se révèle pas la meilleure solution pour assurer la FT des actionneurs. Comme nous l'avons vu plus haut l'actionneur dans ce type de modèle (SAMS) devient un point unique de défaillance « *single point to failure ».* C'est pour cette raison que le groupement de plusieurs clusters SAMS, pour obtenir un seul cluster en MAMS, pourra nous faire profiter de tous les avantages d'un modèle MAMS en termes de fiabilité et de FT, sans pour autant multiplier la taille du cluster en termes de prise de décision. Ceci pourra causer une surcharge ou *overload* de l'actionneur CH qui verra sa charge d'énergie diminuer trop rapidement et sa bande passante saturée par le nombre de capteurs qui le sollicite.

C'est ce cluster en MAMS qui regroupe au moins deux clusters SA que nous appellerons *Cluster Autoréparable* ***CA***. Autoréparable, parce que chaque ***CH*** est partagé entre deux clusters en terme de réaction dans l'environnement. Concrètement, dans un modèle SAA, l'ensemble d'équipements d'actuations ***Oi*** du cluster ***Ci*** sera partagé avec le cluster voisin ***Cj*** et aussi l'inverse ; l'ensemble d'équipements d'actuations ***Oj*** du cluster ***Cj*** est partagé avec le cluster ***Ci***. De cette façon, les deux clusters ***Ci*** et ***Cj*** assurent un backup mutuel l'un pour l'autre et forment un cluster autoréparable que nous appellerons ***CAij***.

En pratique, sur le terrain nous pouvons trouver un ou plusieurs équipements d'actuations dans ***Oi*** qui ne peuvent pas intervenir dans ***Cj***, en raison de leur mobilité d'action limitée. Une condition nécessaire pour la construction d'un cluster autoréparable ***CAij*** entre ***Ci*** et ***Cj*** est que : le sous ensemble ***Oij*** qui contient les équipements d'actuation de ***Oi*** qui peuvent intervenir dans ***Cj*** est non vide pour couvrir ***Wj*** *entièrement* et l'inverse, c.à.d. le sous ensemble ***Oji*** qui contient les équipements d'actuation de ***Oj*** qui peuvent intervenir dans ***Ci*** est aussi non vide pour couvrir ***Wi*** *entièrement*.

Si cette condition nécessaire est remplie par plusieurs clusters voisins de ***Ci***, le choix se fera suivant : la fiabilité, l'énergie consommée et les délais d'une communication Actor-Actor respectivement. Une métrique doit être proposée pour assurer un bon choix de backup pour le cluster ***Ci***.

4.3. Communication tolérante aux défaillances dans un CA :

Utiliser un plan de routage efficace intra et inter clusters qui pourra assurer une QoS, peut vite se révéler très important pour minimiser les délais d'intervention et l'énergie consommée par les messages redondants vers les deux Actionneurs CH. Aussi, une QoS peut être critique pour certains types d'applications qui auront besoin d'un temps de latence minimal tel qu'un système d'extinction d'incendie de forêt qui se propage vite.

Deux types de chemins peuvent être calculés pour satisfaire les diverses exigences suivant le type d'application :

Chemin à temps réel ou *Real-Time Path* RTP: C'est le chemin le plus rapide pour atteindre un actionneur CH, et ainsi, minimiser le temps de latence pour la réaction dans l'environnement. Ce type de chemins peut être gourmand en énergie et peut aussi causer une déconnexion du réseau, c'est pourquoi un nœud qui voit sa charge d'énergie diminuer en dessous d'un certain seuil peut choisir de ne pas router en PRT. Raisonnablement, un temps de latence relativement petit pour un saut vaut mieux qu'une route déconnectée. Une optimisation de la taille des paquets avant l'envoi, avec des schémas de compression plus évolués, est aussi de mise pour minimiser la consommation d'énergie d'une communication en PRT.

Chemin d'énergie minimal ou *Low Energy Path* LEP : C'est le chemin le moins couteux en énergie pour atteindre un actionneur. Il peut être lent mais très utiles puisqu'il économise l'énergie des capteurs. Il trouve son intérêt dans les communications qui n'ont pas de contraintes de temps et dans les communications périodiques. Par exemple, l'envoi de la température d'un équipement toutes les 30minutes peut utiliser un LEP au lieu d'un RTP.

Ceci dit, les deux types de chemins RTP et LEP, même s'ils peuvent permettre une meilleure fluidité des informations au tour des CH, ils n'assurent pas une FT.

Pour assurer une meilleure FT, nous allons envoyer les données captées vers plusieurs actionneurs en même temps ; nous avons appelé ça une communication ***Multi-Destinations***. Plus encore, dans un cluster autoréparable **CA**ij, l'envoi de paquet en ***Multi-Destinations*** vers les actionneurs **CHi** et **CHj** peut permettre de tolérer les pannes des deux actionneurs. En cas de panne de l'actionneur ***CHi,*** l'actionneur ***CHj*** va aussi

disposer de toutes les informations sur les événements ***Ei*** et peut intervenir dans ***Wi*** avec les équipements d'actuations dans ***Oji.***

Quatre types de communications capteurs actionneurs peuvent être définis :

Real Time Multi Destinations (RTMD) : C'est l'envoi des paquets en multi destinations en même temps à travers les chemins à temps réel. Ce type de communication capteur actionneur est très gourmand en énergie et ne doit être utilisé que pour les applications à temps réel qui n'acceptent pas de temps de latence dû à la retransmission des messages perdus.

Real Time Single Destination (RTSD) : Ce type de communication assure l'envoi de paquets d'un événement Ei vers un seul actionneur CHi. Pour permettre d'assurer une FT, le capteur qui envoie l'événement ***Ei*** peut choisir de recevoir un ACK ou pas, suivant l'importance de l'événement et le temps de latence qu'il peut supporter. Dans le cas où l'accuser (*ACK)* demandé par le capteur ***Si*** n'arrive pas après un temps limite, le capteur suppose que le ***CHi*** est en panne et envoie le même événement au ***CHj,*** le Cluster Head du cluster voisin du même cluster autoréparable ***CHij***. Cette communication peut permettre une FT des actionneurs, mais engendre un temps de latence supplémentaire dû au temps limite décompté par le capteur ***Si***. L'avantage est le gain en énergie, si en considère que les pannes d'un CH ne sont pas très fréquentes, et que l'envoi de l'ACK de taille relativement petite par rapport au paquet de données peut nous permettre d'économiser l'énergie consommée par une communication ***RTMD***.

Low Energy Multi Destinations (LEMD) : Ce type de communication pourra être utilisé par des applications qui n'ont pas de contraintes de temps, mais par contre, exigent une FT pour les données envoyées. Cette communication permet une réplication des données sur les CH d'un Cluster autoréparable CAij et assure une reprise après panne par la technique du *checkpointing* qui a été détaillée dans le Chapitre 2.

Low Energy Single Destination (LESD): C'est le type de communication le moins couteux et le moins rapide. Il peut être utilisé pour les transmissions périodiques qui ne nécessitent pas de FT. Par exemple, la récolte des informations de bon fonctionnement d'un système de détection d'incendie de forêt toutes les 30minutes.

Dans une communication Multi Destinations, qu'elle soit *Real Time* ou *Low Energy,* si un CHi reçoit des données d'un événement Ei, il doit accuser l'exécution de la

tache au CHj voisin dans CAij. Dans le cas où le CHj ne reçoit pas d'accusé d'exécution de tache pour l'événement Ei, il va supposer que le CHi est en panne, et de ce fait, il va intervenir dans la zone d'événements Wi pour répondre à Ei avec l'ensemble des équipements d'actuation Oji.

Dans notre travail, nous avons supposé que l'exécution redondante d'une action de la part de plusieurs actionneurs peut être permise, ce qui est effectivement le cas dans certain domaine d'application des WSANs. Dans le cas contraire, des algorithmes d'exclusion Mutuelle existe, pour éviter l'exécution redondante d'une action sur un seul événement.

Un capteur disposera de deux tables de routages : une pour les RTP et l'autre pour les chemin LEP. La table de routage est très réduite, elle ne contiendra que le prochain saut vers le ***CHi*** et ***CHj,*** dans le cas où le capteur appartient à un ***CAij***.

Par contre, une table de routage d'un ***CHi*** contiendra tous les chemins vers les capteurs dans ***Si***. Dans le cas où un ***CHi*** veut joindre un ***CHj*** voisin, il transmet les paquets à un capteur de bordure ou *Sensor Border* ***SBi*** qui appartient à ***Si***. Ce dernier transmet les paquets à un ***SBj*** qui appartient à ***Sj***. Le ***SBj,*** de la même manière qu'un ***Sj,*** saura joindre son actionneur Cluster Head ***CHj***.

Les SBs sont des nœuds avec une fonctionnalité particulière, ils sont responsables de la communication *inter cluster*. Ils doivent maintenir dans leurs tables de routage le prochain saut pour atteindre le cluster voisin d'un même cluster autoréparable.

Dans ce qui va suivre nous allons supposer que :

1. Tous les capteurs d'un WSAN connaissent leurs voisins directs à un saut. Une étape d'initialisation peut être lancée, où chaque capteur du réseau envoie un message « Beacon » pour récupérer ses voisins directs.
2. Les nœuds capteurs et actionneurs d'un WSAN sont synchronisés avec leurs voisins directs à l'aide d'un algorithme de synchronisation tel que présenté dans [BoM07, ChS07].
3. Les actionneurs ou les équipements d'actuations peuvent être mobiles pour couvrir une plus grande zone de réactions.

5. Protocole proposé:

5.1. Principe de fonctionnement :

Nous avons proposé un protocole sur trois (03) phases, qui permet de diviser un réseau de capteurs et d'actionneurs en clusters autoréparables capables de profiter des avantages d'un modèle MAMS, sans pour autant pénaliser les actionneurs CH avec une charge de communication et de coordination supplémentaire. L'algorithme de clustering est simple, rapide et permet d'obtenir des clusters de tailles égales. Il permet également d'établir un plan de routage avec une QoS qui assure une fluidité des données, pour répondre à l'exigence des applications à temps réel et tolérer les pannes des actionneurs comme celles des capteurs.

Dans ce qui va suivre, nous estimerons que le cluster qui transmet le HELLO en cours de traitement est le cluster source, et que ses clusters voisin sont des clusters destination.

5.1.1. Phase 1 (HELLO):

Un actionneur ***CH-source*** qui arrive dans une zone d'événements ***W-source***, envoie un message **HELLO** en *broadcast* avec un **TTL** choisi auparavant, suivant la quantité d'énergie de l'actionneur, le nombre de ses voisins directs et ses capacités de calculs et de transmissions.

Chaque capteur libre ou Sensor Free **(SF)** qui reçoit un **HELLO,** utilise les informations sur le chemin pour calculer les chemins RTP et LEP, en utilisant la métrique suivante :

$$\textbf{Routing\_metrique} = \begin{cases} \sum_{I \in path} \textbf{Delay}_I & \textbf{If Real Time Path} \\ \sum_{I \in path} \textbf{Energ}_I & \textbf{If Low Energ Path} \end{cases}$$

Delay $_I$: temps de vol du message pour chaque saut **I**.

Energ $_I$: l'énergie consommait pour chaque saut **I**.

Les tables de routages RTP et LEP seront mises à jours de façon à pouvoir atteindre le nœud actionneur qui à transmit le **HELLO**. De ce fait, le **SF** devient un **S-source** qui appartient au cluster Source. Le **S-source** vérifie si le compteur **HOP** est inférieur au **TTL** et broadcast, à son tour, le message **HELLO** après avoir incrémenté le compteur de saut **HOP.** La mise à jours des champs ID_S, Energ, Energ_Hop, Time et Send_Time est faite par le capteur avant l'envoi du paquet.

Si le message **HELLO** est reçu par un capteur qui appartient déjà à un autre cluster voisin **CH-destination,** ce capteur devient un capteur de bordure ou Sensor Border **(SB).** Le **SB** sera responsable du lancement de la deuxième phase de protocole Figure 17. Le message **HELLO** aura la forme suivante :

ID_CHs	**W_CHs**	**HOP**	**TTL**	**ID_S**	**Eneg**	**Energ_Hop**	**Time**	**Send_Time**

ID_CH : Identificateur du nœud actionneur à l'origine du HELLO
W_CH : La zone d'actuation couverte par le CH.
HOP : Compteur de nombre de saut réaliser par le HELLO
TTL : Nombre de saut limite à réaliser par le HELLO.
ID_S: Identificateur du dernier capteur qui fait transiter le HELLO.
Energ: Le cumul d'énergie consommée le long du chemin du HELLO.
Energ_Hop: L'énergie consommée lors du prochain saut à effectuer par le HELLO.
Time: Le cumul du temps d'envoi du HELLO le long du chemin du message.
Send_Time: L'heure à laquelle le message a été envoyé, il servira à calculer le temps de vol du message HELLO.

Les champs **ID_S**, **Energ**, **Energ_Hop**, **Time** et **Send_Time** servent à calculer les chemins temps réel ou Real Time Path (**RTP**) et les chemins à consommation d'énergie minimum ou Low Energie Path (**LEP**).

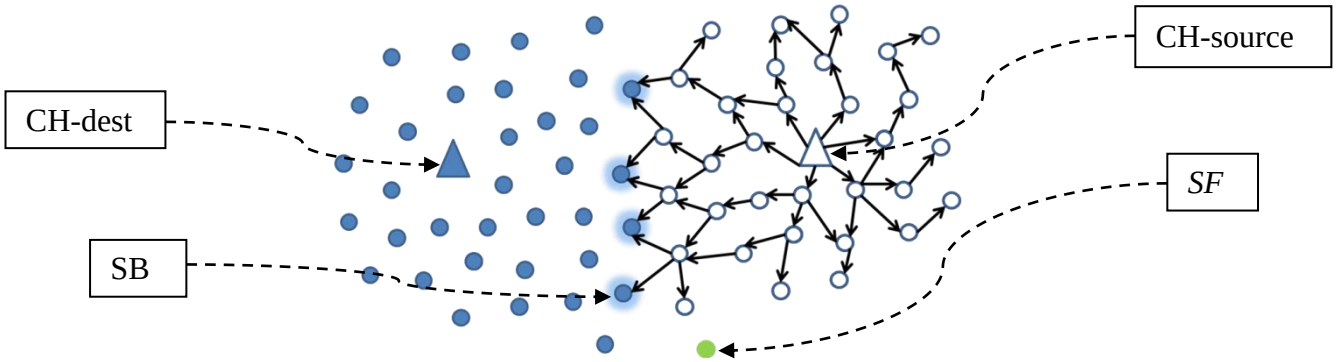

Figure 17 : Envoi du HELLO avec un TTL=5

5.1.2. Phase 2 (CA_REQ):

Après avoir reçus un **HELLO** de tous ces voisins directs, ou après un temps limite, chaque **SB** envoie un message **CA_REQ** à son actionneur primaire **CH-destination** avec les informations sur la zone d'actuation **W-source** (zone d'actuation couverte par **CH-source**). Ces informations serviront au **CH-destination** pour vérifier la condition nécessaire et suffisante pour la construction d'un cluster autoréparable entre les deux clusters source et destination, autrement dit si :

W-dest = W-source.

Si un capteur **S-dest** qui appartient au cluster destination reçois un **CA_REQ** il le forward vers le **CH-dest** à travers le chemin RTP calculé dans la première phase du cluster destination Figure 18.

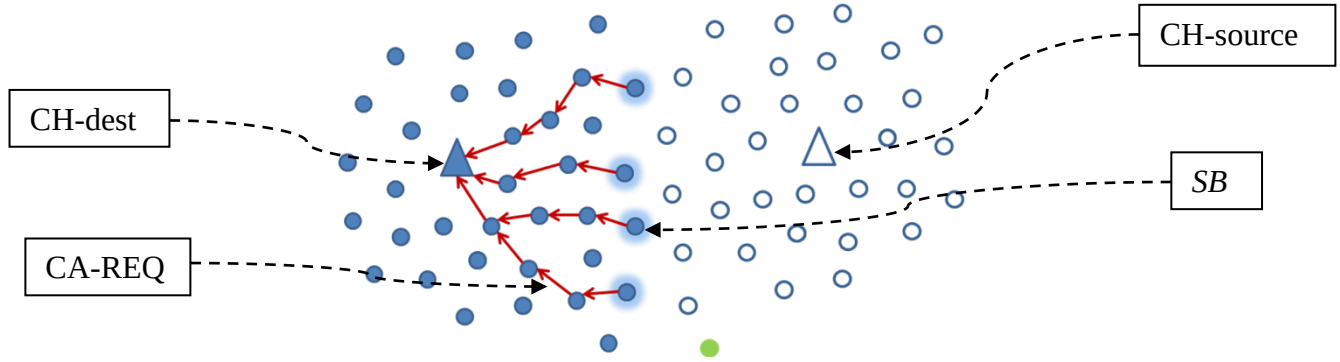

Figure 18 : Les SB envoi un **CA_REQ au CH-dest**

Suivant le résultat de la condition de construction d'un CA, le **CH-dest** répond par un **CA_REP** positif ou négatif mais uniquement aux SB qui sont choisies comme

passerelle ou « Sensor Gateway » (**SG**) de communication entre les deux clusters source et destination, le choix des gateway se fait suivant la métrique suivante :

SG-metrique = Energ_SB / Load_SB

Energ_SB : L'énergie résiduelle du **SB** à l'envoi du message **CA_REQ**.

Load_SB : Le nombre de voisins directs du capteur **SB**.

De cette façon, les deux meilleures gateway les plus durables seront choisis. Le choix de limiter à deux les **SG** est justifié par le fait que les avantages d'une triple redondance de gateway sont négligeables par rapport à la perte d'énergie engendré lors de la 3ème phase du protocole.

Le message **CA_REP** empruntera le chemin RTP inverse vers les deux **SG**, ces derniers seront chargés du lancement de la 3ème phase Figure 19.

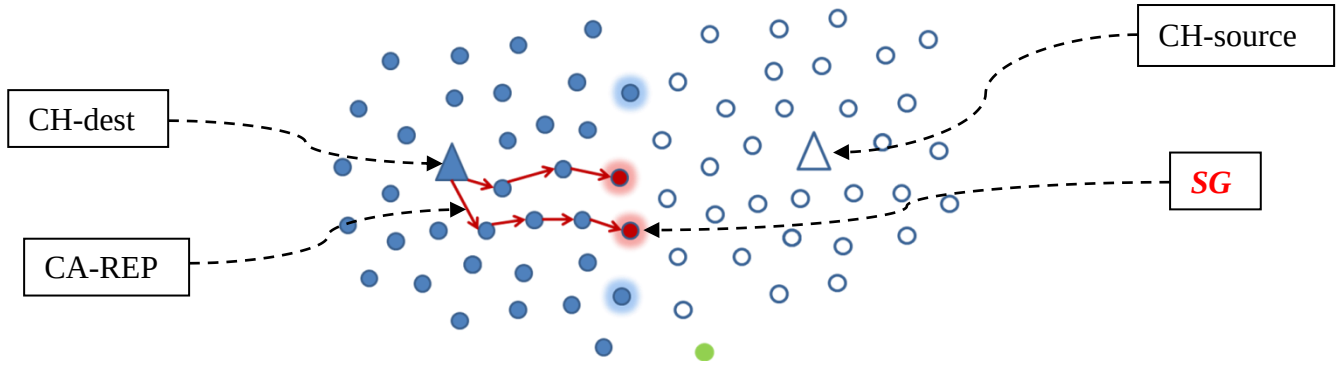

Figure 19 : l'actionneur **CH-dest** réponde avec **CA_REP**.

Le message **CA_REQ** aura la forme suivante :

ID_S	ID_CH-dest	ID_CH-source	W_CH-source	SG_Métrique

SG_Métrique: Métrique pour le calcul des meilleures Gateway.

Le message **CA_REP** aura la forme suivante :

ID_S	REP

5.1.3. Phase 3 (Route Update) :

Les SB se comportent de façon différente suivant la réponse transmise dans le **CA_REP**. Si le **CA_REP** est négatif les deux **SG** transmettent juste un **HELLO_REP** négatif en unicast vers le **CH-source.** Figure 20, ce qui signifie que, le CA entre le cluster source et destination ne peut pas être construit en raison de la non satisfaction de la condition nécessaire, autrement dit : **W-dest != W-source**.

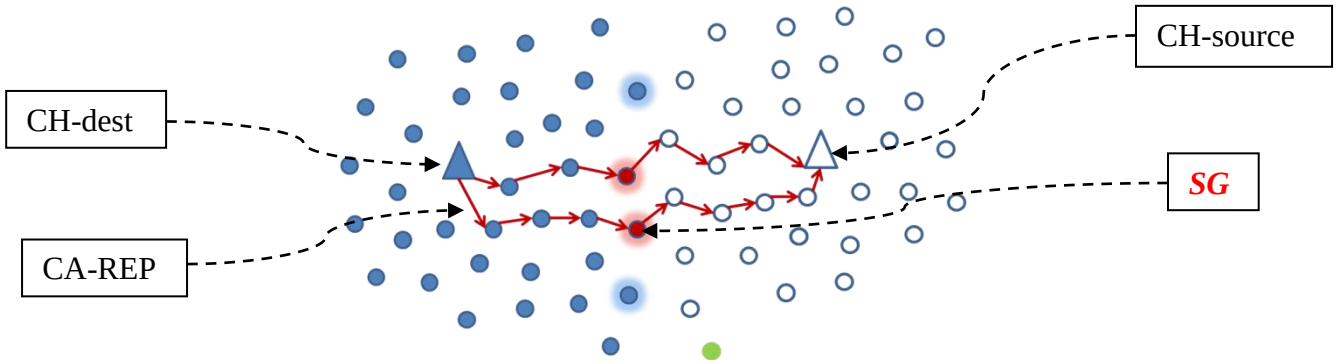

Figure 20 : les **SG** transfert un **CA_REP** négative.

En revanche, si le **CA_REP** est positif les deux **SG** choisis, diffuse le **HELLO_REP** avec un TTL= 2*TTL de façon à atteindre tous les capteurs **S-source,** mais aussi, les **S-dest** des clusters source et destination et mettre leurs tables de routage RTP et LEP à jours. Ainsi, les **S-source** et **S-dest** seront capables de joindre les deux **SG** qui relieront les données entre les deux clusters. Figure 21.

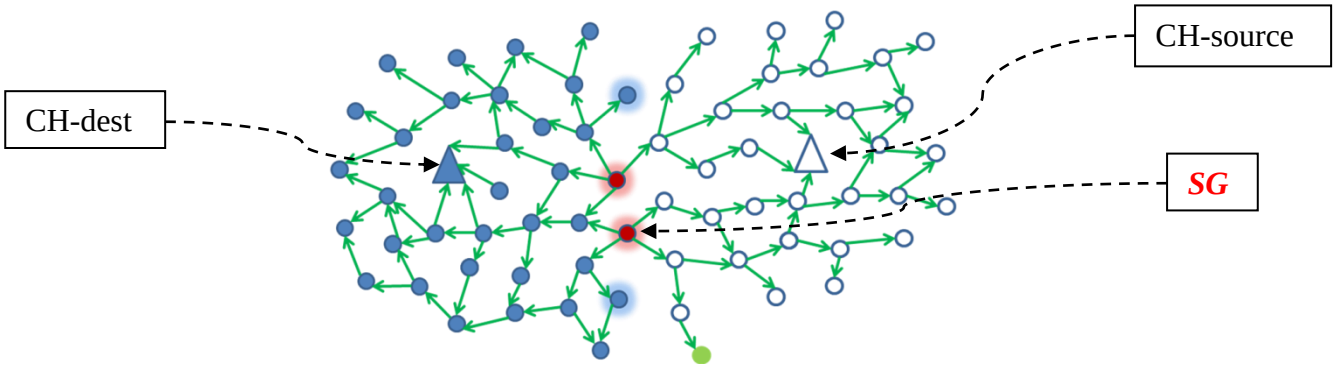

Figure 21 : les **SG** MAJ les tables de routage de tous les capteurs dans CA

Chaque capteur qui appartient au cluster source devra avoir deux tables de routage RTP et LEP, qui dispose de deux chemins qui pointent vers le **CH-source,** et deux autre chemins qui pointent vers les deux SG qui devront relier les communications

vers le cluster destination, et vice versa Figure 21. Un message **HELLO_RESPONSE** aura la forme suivante :

ID_S	ID_CH-source	ID_CH-dest	Eneg	Energ_Hop	Time	Send_Time

ID_S: Identificateur du capteur qui envoie le HELLO_RESPONSE.
Energ: le cumul d'énergie consommée le long du chemin du HELLO_RESPONSE.
Energ_Hop: L'énergie qui va être consommée dans l'envoi du message HELLO_RESPONSE.
Time: le temps d'envoi du HELLO_RESPONSE le long du chemin du message.
Send_Time: l'heure à laquelle le message a été envoyé, il servira à calculer le temps de vol du message HELLO_ RESPONSE.

5.1.4. Transmission des données :

Après avoir mis à jour les tables de routage des capteurs **S-source** avec les chemins RTP et LEP, chaque capteur de la zone d'événements **W-source** saura joindre le CH-source et le CH-dest à travers les deux **SG** qui assurent la communication *inter-cluster*.

La transmission des données dans un cluster autoréparable se fait avec quarte (04) modes différents suivant la valeur de qualité de service (QoS) :

Multi-Actors Real Time Path (**MA-RTP)**: Un capteur **S-source** transmet la donnée aux deux actionneurs du CA sur les chemins RTP. Ce type de transmission est le plus fiable et le plus rapide. Il est utilisé pour les données critiques, où le temps de réaction doit être le minium possible. Un actionneur de destination **CH-dest,** en recevant une donnée avec une QoS=1 venant d'un **S-source**, attend un temps limité, si le **CH-dest** ne reçoit pas l'ACK de bonne exécution de la tache de la part du **CH-source,** il réagit directement sur l'événement à la place de l'actionneur source **CH-source**. Le problème d'exclusion mutuelle n'est pas pris en charge dans notre protocole.

Multi Actors Low Energie Path (**MA-RTP)**: Un capteur **S-source** transmet la donnée aux deux actionneurs du CA sur les chemins **LEP**. Ce type de transmission est fiable et sans contrainte de temps. Il est utilisé pour les données où la réaction est

critique, même si elle n'est pas à temps réelle. L'actionneur de destination **CH-dest,** qui reçoit une donnée avec une QoS=3 venant d'un **S-source**, attend un temps limite, si le **CH-dest** ne reçoit pas l'ACK de bonne exécution de la tache par le **CH-source**, il retransmet la donnée au **CH-source** une deuxième fois si le **CH-source** n'accuse pas la bonne exécution de l'action, le **CH-dest** réagit sur l'événement à la place de l'actionneur **CH-source** après un temps d'attente limité.

Les deux modes de transmission *Single Actor RTP* (**SA-RTP**) et le *Single Actor LEP* (**SA-LEP**) sont plus exposés aux pannes de liens et d'actionneurs, mais ils permettent une économie dans la consommation d'énergie par rapport au deux premier mode.

Un CH-dest peut décider suivant la criticité de l'événement, signalé par un champ QoS qui accompagne chaque paquet de données, soit de retransmettre la donnée après le temps T soit d'intervenir directement dans *W-source* et accuser son intervention à l'actionneur CH-source. Figure 22.

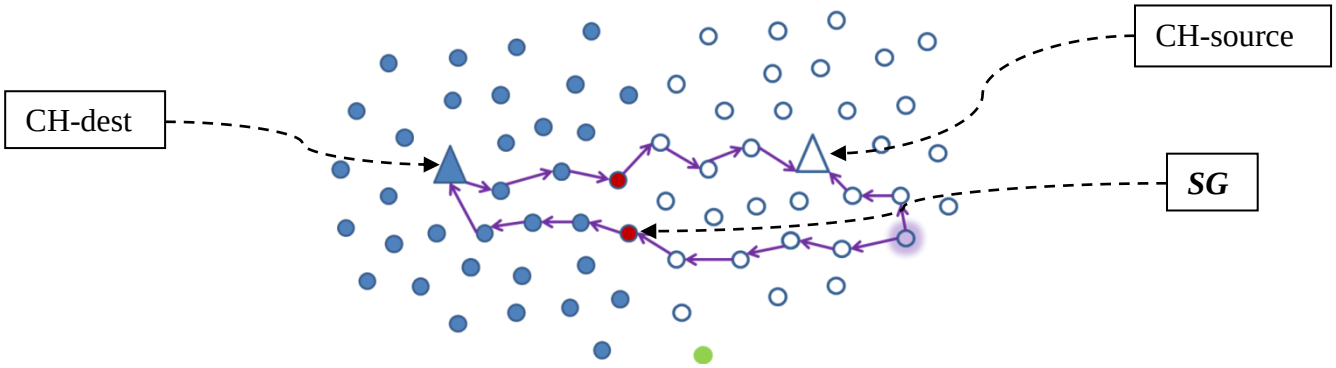

Figure 22 : Communication tolérante aux défaillances des actionneurs

5.2. Constantes et variables:

Chaque nœud possède les constantes et variables suivantes :

Sensor : structure qui contient tous les informations sur un capteur (ID, ID_CH-dest, ID_CH-source, …)
CH : structure qui contient toutes les informations sur un CH (ID, AREA,…)
SG_metrique : représente l'énergie résiduelle d'un SB divisée par le nombre de ses voisins.

msg_HELLO : (ID_CH-src, W_CH-src, HOP, TTL, ID_S, Energ, Eneg_Hop, Time, Send_Time)
msg_CA_REQ : (ID_S, ID_CH-dest, ID_CH-src, W_CH-src, SG_Métrique)
msg_CA_REP : (ID_S, REP)
msg_HELLO_REP : (ID_S, ID_CH-src, ID_CH-dest, Energ, Eneg_Hop, Time, Send_Time)
msg_DATA : (ID_S, QoS, DATA)
WaiteForHello : délai utilisé après la réception d'un HELLO. C'est pour attendre la découverte de plusieurs chemins. Ce délai doit être aussi petit que possible afin de ne pas influencer le délai de construction du cluster autoréparable CA.

RTP[] : la table de routage qui contient le prochaine saut dans un chemin à temps réel vers le CH-dest d'un CA.

LEP[] : la table de routage qui contient le prochaine saut dans un chemin à consommation minimale vers le CH-dest d'un CA.

5.3. Evénement :

Chaque nœud Capteur S traite les événements suivant :

When receive HELLO :

Cet événement est interprété différemment suivant le type de capteur. Le capteur libre SF qui reçoit un HELLO calcule les chemins RTP et LEP pour atteindre l'actionneur CH-source qui a envoyé le HELLO et broadcast ce dernier si le nombre de sauts limités par le TTL n'est pas atteint.

Dans le cas où le HELLO rencontre un capteur qui appartient à un autre cluster (CH-dest), ce capteur est déclaré comme SB et attend un délai minimal fixé par WaiteForHello, avant qu'il ne transmet un CA_REQ à son actionneur primaire CH-dest.

When receive CA_REQ:

Les capteurs ne traitent pas le CA_REQ ils le forward juste à leurs actionneurs primaires. Ce dernier décide si le CA peut être construit, et choisit les SG pour la communication inter cluster en leur transmettant un CA_REP.

When receive CA_REP :

Cet événement est traité différemment suivant le type de capteur et la provenance du message. Un capteur S-source qui reçoit un CA_REP venant de CH-source le transfert à travers le chemin RTP au SG choisi par le CH-source. Si le CA_REP est négatif les SG envoi la réponse en unicast au CH-source, sinon il broadcaste un HELLO_REP à tous les capteurs présents à 2*TTL des SG pour mettre à jours leurs tables de routage et informer l'actionneur CH-source que le CA est construit.

When receive HELLO_REP :

Cet événement est traité de la même manière qu'un HELLO s'il est positif, sinon un simple unicast de notification vers le CH-Source pour dire que cluster AUTO réparable ne peut pas être construit.

When receive a data packet to forward:

Cet évènement est déclenché quand un nœud reçoit un paquet de données à transmettre vers un actionneur, ou quand lui-même est la source du paquet et veut le faire parvenir à un CH ou aux deux CH du CA. Le paquet de donnée est accompagné d'un champ QoS qui permet de savoir quelle est la destination de ce dernier.

5.4. Algorithme :

Initialization phase

```
If (Node is Actor) Broadcast HELLO;
```

When receive HELLO :

```
If      (Node is SF)
        Calculate RTP and LEP to CH_Source and Update route Table ;
        If   (Hop< TTL)
                Hop++;
                Update and Broadcaste HELLO;
        EndIf
Else    Node = SB;
          WaiteForHello ;
          Initialize and Send CA_REQ to CH_dest;
EndIf
```

When receive CA_REQ:

```
If  (Node is Actor)
        Select best two SG;
        If (CA condition is TRUE)        Initialize and Send CA_REP to SG with RESPONSE=TRUE

        Else   Initialize and Send CA_REP to SG with RESPONSE=FALSE
        EndIf
Else   Update and Forward CA_REQ to CH_dest
EndIf
```

When receive CA_REP

```
If (Node is Sensor)
        Update and Forward HELLO_REP to SG
        If  (Node is SG)
            If   (CA_REP.RESPONSE is TRUE)
                    Node = SG;
                    Update and Broadcast HELLO_REP with RESPONSE= TRUE;
            Else   Update and Forward HELLO_REP to CH_source with RESPONSE= FALSE;
            EndIf
        EndIf
EndIf
```

When receive HELLO_REP

```
If (CA_REP.RESPONSE is TRUE)
        Calculate RTP and LEP to SG and Update route Table;
        If   (Hop< TTL)
                Hop++;
                Update and Broadcaste HELLO_REP with RESPONSE=TRUE;
        EndIf
Else   Update and Forward HELLO_REP to CH_Source with RESPONSE = FALSE;
EndIf
```

When receive data to forward:

```
Switch DATA.QoS :
        CAS 1: Use RTP to send data   CH-source and CH-dest; Beak;
        CAS 2: Use RTP to send data to CH-source only; Beak;
        CAS 3: Use LEP to send data to CH-source and CH-dest; Beak;
        CAS 4: Use LEP to send data to CH-source only; Beak;
```

6. Conclusion

Une tolérance aux fautes est un compromis entre le cout et la fiabilité. Notre protocole de clustering reste rapide et local. Il permet d'obtenir des clusters autoréparables ***CA*** de taille plus ou moins égales et chaque cluster contient deux actionneurs CHi et CHj. Le protocole de clustering peut s'exécuter périodiquement, si la topologie du réseau change.

La gestion d'un cluster *Ci* en interne incombe au *CHi.* Dans un cas de panne d'un *Chi,* le *CHj* voisin membre d'un même CA peut le remplacer dans toutes ses taches de décision et d'actuation. Que nous soyons dans un schéma ***SA*** ou ***SAA,*** le protocole fonctionne de la même manière, sauf pour la contrainte de l'exclusion mutuelle. Cet aspect n'a pas été traité dans ce travail. L'exclusion mutuelle entre les éléments de *Oi* a été discutée dans [NaT94, VZS06].

Nous avons aussi proposé un protocole de routage qui permet quatre types de communication qui peuvent être utilisés suivant le domaine d'application des WSANs. Les communications sont classées dans deux catégories principales à savoir : les communications à temps réel pour les applications qui ne supportent pas un temps de latence, et les communications à consommation d'énergie minimale pour les applications à prélèvements périodiques.

Dans ce qui va suivre, nous allons simuler notre protocole sous TOSSIM, pour démontrer son efficacité et sa scalabilitée par rapport à d'autres protocoles qui traitent de la tolérance aux défaillances des actionneurs dans les WSANs.

Chapitre IV : Environnement, démarche et résultat des simulations.

1. Introduction :

Avant sa mise en place, le déploiement d'un WSN nécessite une phase de simulation afin de s'assurer du bon fonctionnement de tous les protocoles de communication qu'il utilise. En effet, pour de grands réseaux, le nombre de capteurs peut atteindre plusieurs milliers et entraine donc un coût financier relativement important. Ainsi, il faut réduire au maximum les erreurs de la conception.

Pour évaluer les performances du protocole que nous avons proposé dans le Chapitre précédent, et le comparer avec le protocole qui se base sur la coordination semi passive SPC pour répondre à la problématique de tolérance aux pannes dans un modèle MAMS, nous avons effectué des simulations qui valide et montre les avantages apportés par notre protocole par rapport au deux modèles SAMS et MAMS-SPC. Avant de parler des simulations, nous allons d'abord présenter, l'environnement de simulation utilisé. Il s'agit de simulateur TOSSIM [LLW03] qui est le simulateur de TinyOS [TINY], développé en langage Nec.

Ce Chapitre est composé, principalement, des parties suivantes : après cette brève introduction, nous allons donner quelques notions générales sur le système d'exploitation des capteurs le plus utilisé à savoir TinyOS. Nous allons aussi présenter la plateforme de simulation TOSSIM et montrer ses principaux avantages. Après cela, nous allons détailler le choix des critères d'évaluations et les scénarii adoptés pour les simulations. Nous terminerons par les résultats de simulation avant de conclure ce Chapitre.

2. TinyOS :

Suite aux différents défis des réseaux de capteurs, les chercheurs de l'université de Berkeley, en plus de nombreux contributeurs ont développé un système d'exploitation destiné aux réseaux de capteurs afin de faciliter l'implémentation et l'exécution de protocoles 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 est intitulé TinyOS.

Il 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. Il utilise une programmation entièrement réalisée en langage NesC.

2.1. Propriété de TinyOS :

Les systèmes d'exploitation pour les nœuds capteurs sont généralement moins complexes que les autres systèmes. Plusieurs systèmes d'exploitation ont été proposés pour les WSN parmi lesquels on trouve SOS [CEM05], Contiki [ABT04], MANTIS [CCJ06]. TinyOS reste néanmoins le plus répandu pour les WSNs car il répond aux exigences particulières des applications des WSNs. Il convient alors de mentionner les propriétés qui rendent TinyOS aussi populaire pour ce genre de réseaux:

- 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.

2.2. Principe général :

TinyOS est construit autour des différents concepts décrits ci-dessous: [SYL07]

- Les composants : qui sont eux même 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 permettent de spécifier des fonctions, des commandes ou des événements. Ces fonctions sont alors implémentées par le fournisseur ou l'utilisateur de l'interface.

2.3. Langage NesC

NesC est un langage de programmation orienté composant syntaxiquement proche du langage C. Il est conçu pour la réalisation des systèmes embarqués distribués, en particulier, les WSN [TINY].

Il existe trois types de fichiers sources des applications NesC, les fichiers interfaces, les fichiers configurations et les modules qui constituent les composants [DPH06].

Un fichier de configuration définit les composants et les interfaces utilisées par l'application déployée sur le capteur. Il définit aussi la description des liaisons entre eux.

Un module constitue la brique élémentaire du code et implémente une ou plusieurs interfaces. Une interface définit d'une manière abstraite les interactions entre deux composants. Elle définit un fichier décrivant les commandes et les évènements proposés par le composant qui les implémente. Une commande doit être implémentée par le fournisseur de l'interface et un évènement doit être implémenté par l'utilisateur de l'interface.

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. [YAC08]

3. Simulateur de TinyOS : TOSSIM

3.1. Présentation :

Pour arriver à simuler le comportement des capteurs au sein d'un WSN, les chercheurs de l'université de Berkeley ont développé un outil de simulation très puissant pour TinyOS sous le nom de TOSSIM.

TOSSIM est un simulateur d'événement discret, en comparaison avec les autres simulateurs tel que NS-2, TOSSIM capture le comportement et les interactions des réseaux de capteurs non pas à un niveau de paquet mais aussi à un niveau de granularité très bas.

Le principal but de TOSSIM est de créer une simulation très proche de ce qui se passe dans les WSN dans le monde réel. Une économie d'effort et une préservation du matériel sont possibles grâce à cet outil.

3.2. Architecture générale:

L'architecture de TOSSIM est composée de cinq (05) éléments de base :

- Le diagramme de composant TinyOS ou Frame.
- Les événements ou modèle d'exécution.
- Les modèles Radio et ADC (Analog-Digital-Conversion).
- L'émulation des composants Hardware.
- Les services de communication.

Chaque programme TinyOS est un diagramme de composants. Chacun des composants de TinyOS représente une entité de calcul indépendante. La première partie de l'architecture de TOSSIM est le diagramme de composants ou Frames, qui permet à TOSSIM de profiter des avantages de la structure de programme TinyOS.

La deuxième partie de l'architecture est le modèle d'exécution, qui est essentiel pour la pile d'événements discrets. Il permet d'empiler et de dépiler les événements. Il est aussi responsable du déclenchement de la simulation.

TOSSIM fournit aussi deux déférents types de modèles. Le modèle Radio pour traiter tous les aspects de transmission et le modèle ADC Analog-Digital-Converter. Plus de détails sur ces fonctionnalités sont disponibles [MiH05].

TinyOS résume tous les ressources hardware dans les composants. Une autre partie de l'architecture de TOSSIM contient l'implémentation des composants hardwares de TinyOS.

A la fin, le mécanisme de simulation TOSSIM fournit aussi un ensemble de services de communication pour l'interaction avec les applications externes. Ces services permettent à un programme de se connecter à TOSSIM à travers une socquette TCP pour suivre ou administrer une simulation qui tourne.

4. Choix des critères d'évaluations :

Dans notre simulation nous avons évalué notre protocole par rapport à deux axes importants à savoir l'efficacité et la scalabilitée. Nous avons calculé le taux de réussite d'exécution des tâches qui n'est autre que le nombre de taches réellement exécutées par rapport au nombre de taches lancées. Nous avons aussi calculé la latence d'exécution des tâches qui représente le temps écoulé depuis le déclanchement de l'événement au lancement de la tache à exécuter sur l'événement. Nous avons calculé ces deux paramètres en faisant évoluer le taux d'erreur et le nombre de nœuds, comme nous allons le décrire dans les deux scénarii de simulation par la suite.

Afin de ressortir les avantages de notre protocole nous l'avons comparé à deux autres protocoles que nous avons implémentés. Le premier est le protocole standard sans tolérance aux défaillances dont le fonctionnement est très simple : un nœud donné qui capte un événement ***E*** envoie la donnée à l'actionneur le plus proche. Si ce dernier n'est pas en panne il répond en exécutant la tache sur l'événement ***E***.

Le deuxième protocole avec lequel nous nous somme comparés est le protocole qui se base sur le model MAMS avec une coordination SPC. Le principe de fonctionnement de ce dernier est présente dans le Chapitre deux. Rappelons que ce protocole se base en grande partie sur l'élection d'un seul actionneur primaire actif et qui doit répondre à tous les requêtes d'événement. Tous les autres actionneurs sont donc passifs et ne font que transmettre la donnée reçue vers l'actionneur primaire. En cas de panne de ce dernier, un algorithme de consensus est lancé pour élire un autre actionneur primaire. Nous avons implémenté une version simplifiée que nous avons appelée « SPC-LIKE » pour montrer les avantages de notre algorithme par rapport à ce dernier.

5. Scénarii de simulation :

Dans ce qui va suivre nous allons détailler les scénarii adoptés pour la simulation de notre protocole SR-Cluster mais aussi pour les autres protocoles de comparaison décrit précédemment à savoir SA et SPC-Like.

5.1. Scénario 1 :Taux d'erreur des actionneurs.

Pour le premier scenario nous avons utilisé une grille de 150 nœuds (10*15) parmi eux 10 actionneurs. Afin de tester l'efficacité de notre protocole par rapport au taux de panne. Nous avons fait évoluer le taux d'erreur des actionneurs de 10% à 50% par palier de 10%. Nous avons exécuté la simulation 10 fois pour chaque taux d'erreurs et dans chaque exécution nous avons calculé le taux de réussite des taches et le temps de latence d'exécution des taches. Nous avons aussi calculé les barres d'erreurs avec un intervalle de confiance 95% pour nous permettre d'encadré l'incertitude sur les valeurs des simulations

Pour le choix de la communication, nous avons opté pour la communication RTMD real time multi-destinations qui devra nous permettre d'obtenir des meilleurs résultats concernant les temps de latences d'exécutions des taches.

Ce scénario devrait nous montrer le gain en taux de réussite apporté par notre algorithme en termes d'efficacité d'exécution des taches comparé au protocole standard, ainsi qu'au protocole basé sur SPC que nous avons implémenté. Montré la réduction des temps de la latence d'exécution des taches par rapport au protocole basé sur SPC est aussi un objectif pour ce scénario.

5.2. Scénario 2 : scalabilité de nombre de nœuds.

Dans ce scénario nous avons fixé le taux d'erreurs des actionneurs à 40% et nous avons fait évoluer le nombre de nœuds suivant des grilles de 5*5, 10*5, 10*10, 15*10, 15*15 et 15* 20 en faisant évoluer le nombre d'actionneurs respectivement de 2, 4, 6, 10, 12 et 15. Nous avons de même exécuté la simulation 10 fois pour chaque grille, et nous avons calculé le taux de réussite des taches et le temps de latence d'exécution des taches.

Ce scénario nous permettra principalement de montrer la scalabilitée de notre protocole par rapport aux protocoles de comparaison et surtouts par rapport au protocole basé sur SPC avec un seul actionneur actif pour tout le réseau.

6. Résultats de la simulation :

Nous avons exécuté le premier scénario comme décrit précédemment. Le résultat pour les trois protocoles est représenté sou forme de courbe dans les deux premiers graphs représenté par les Figure 23 et Figure 24.

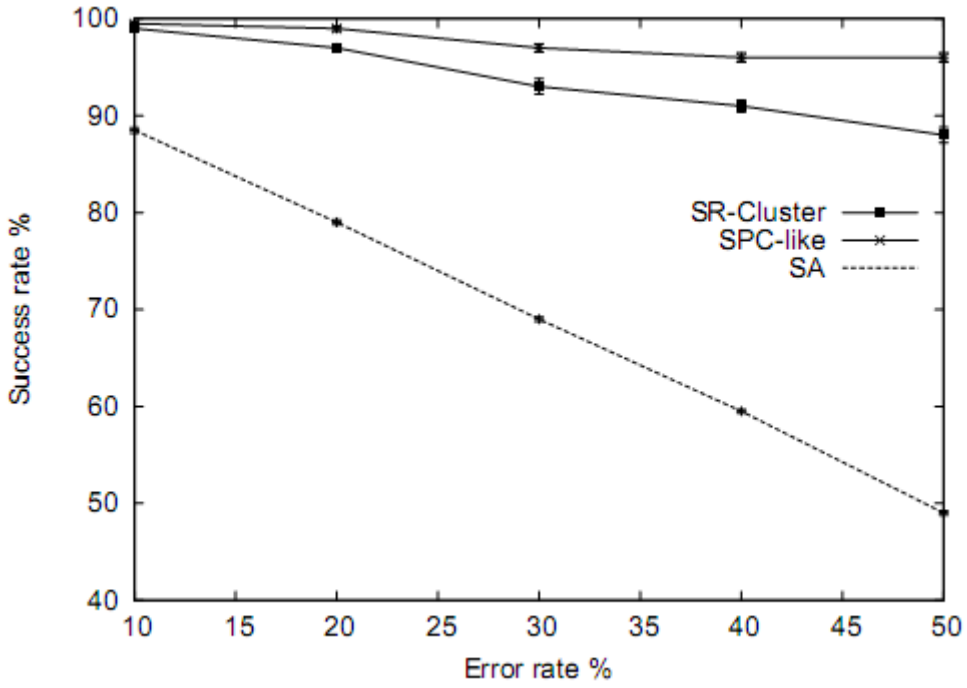

Figure 23 : évaluation du taux de réussite de réaction par rapport au taux d'erreur des actionneurs

Le premier graphe Figure 25 nous permet de comparer l'efficacité de notre algorithme en termes d'exécution d'action. La courbe SA représente un protocole standard sans tolérance aux défaillances. Il est tout à fait normal que la courbe soit linéaire déclinant à 50% de taux de réussite quand le taux d'erreurs des actionneurs augmente jusqu'à 50%.

La simulation a démontré aussi que, même avec un taux de panne d'actionneurs de 50%, notre protocole de clustering qui nous permet de regrouper les clusters en Cluster-Autoréparable (CA), obtient un taux de réussite d'exécution de l'action de 88%. Cela

s'explique principalement par le fait que chaque nœud envoie ses données à deux CH en même temps et quand le premier est en panne (aléatoirement) le deuxième prend le relais et réagit à l'événement. Plus le taux d'erreur évolue, plus la probabilité que les deux actionneurs CH d'un CA seront en panne tous les deux en même temps est grande, et de ce fait, pas de réaction pour ce CA.

Nous remarquons aussi que le protocole SPC-Like procure un taux de réussite minimal de 96% et cela est dû principalement au consensus dont tous les actionneurs sont impliqués. Même avec un taux de panne de 50%, les actionneurs en activité peuvent toujours élire un Actionneur pour réagir à l'événement. La baisse de pourcentage s'explique par la perte des paquets due aux collisions ou un échec lors du processus de consensus.

La Figure 24, nous permet de comparer les temps d'exécution des actions en faisant évoluer le taux de panne d'actionneurs. Nous remarquons que l'algorithme, qui ne permet pas d'avoir une tolérance aux pannes d'actionneurs (SA), s'exécute beaucoup plus rapidement que les autres algorithmes et que le temps d'exécution est plus au moins stable quel que soit le taux de panne des actionneurs.

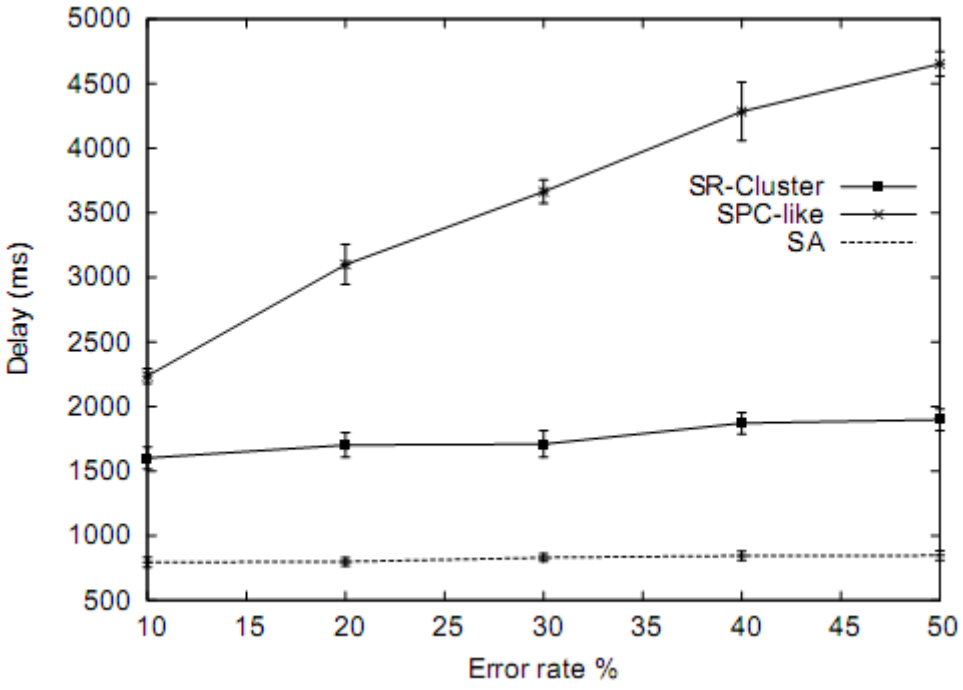

Figure 24 : évaluation du temps d'exécution des actions par rapport au taux d'erreur des actionneurs

La différence de temps de latence entre notre protocole (SR-Cluster) et le protocole (SA) est dû principalement aux délais de réception de l'événement par le CH backup et l'attente d'un ACK de bonne exécution de l'action qui vient naturellement de la part du CH primaire d'un CA. En cas de non réception d'un ACK, le CH backup déclenche une réaction sur l'événement. On remarque que la différence de temps de latence entre les deux protocoles est moyennement constante. Par contre, ce n'est pas le cas pour le protocole SPC-Like qui rentre dans un processus de consensus à chaque fois qu'un actionneur primaire est en panne pour élire un autre actionneur backup à la place. L'exécution de ce consensus consomme du temps et de l'énergie. La tendance à la hausse de la courbe SPC-Like s'explique par le fait qu'à chaque fois que nous perdons un actionneur un processus de consensus est lancé entre les actionneurs restants pour élire l'actionneur Backup. Dans ce cas, un seul consensus est exécuté à la fois, ce qui donne une courbe presque linéaire représentant l'exécution des consensus en série et les temps de latence sont cumulables.

Après l'exécution du deuxième scénario nous avons aussi pu démontrer la scalabilitée de notre protocole par rapport au taux de réussite et le temps de latence comme indiqué par la Figure 25.

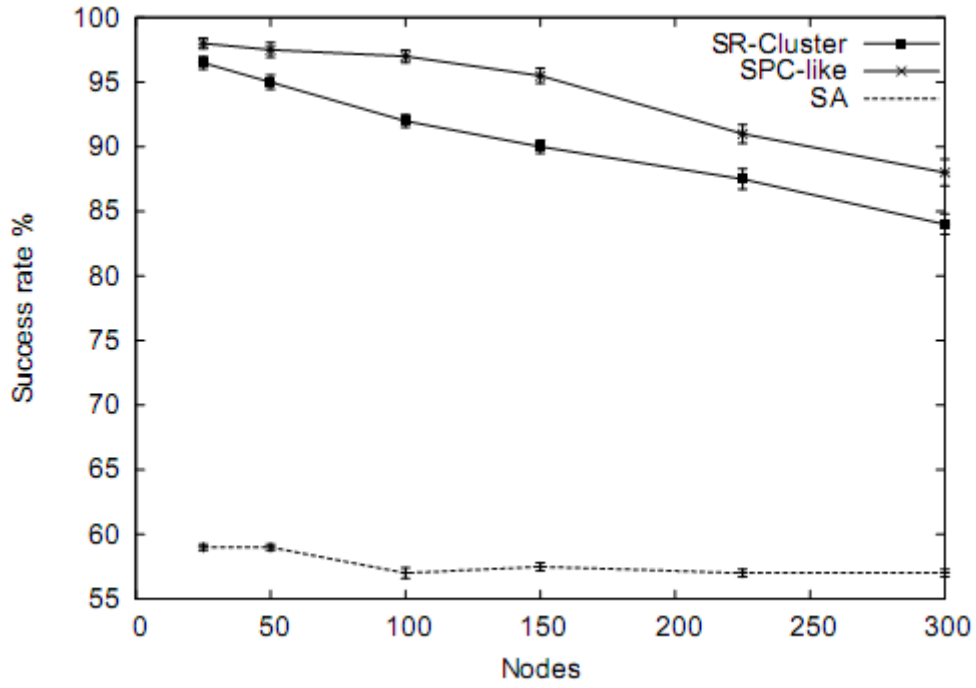

Figure 25 Taux de réussite vs Nombre de nœuds

Nous remarquons que le taux de réussite des deux protocoles SR-Cluster et SPC-Like reste au-dessus de 80% même avec un nombre de nœuds égale à 300 nœuds. Cela montre l'efficacité de notre protocole de clustering qui a réussi à faire face aux pannes des actionneurs en envoyant la donnée captée aux deux actionneurs CH d'un CA qui ont une probabilité de panne de 40%. De ce fait, la probabilité de panne des deux actionneurs d'un même CA en même temps est minime, ce qui donne des taux de réussite assez élevés. Le fait aussi que les communications entre la source et les deux CH d'un CA se font sur des chemins complètement distincts, représente une tolérance aux pannes des capteurs tout au long du chemin (panne de liens).

Pour le protocole SPC-Like, la perte de 40% des actionneurs ne semble pas affecter les taux de réussite qui restent supérieurs à 95% pour les grilles de 25 à 150 nœuds. Par contre, avec les grilles de 225 et 300 nœuds, il enregistre une légère baisse qui peut être due aux collisions et à la perte de paquet de données captées, voire aussi l'échec du processus de consensus entre un nombre important d'actionneurs.

La dernière courbe SA représente un protocole standard sans tolérance aux pannes et de ce fait, elle montre un taux de réussite d'exécution des actions stationnaires de 60%, voire un peu moins quand les réseaux atteignent les 300 nœuds et cela pour un taux de panne des actionneurs de 40%.

La scalabilitée des temps de réponse est aussi démontré pour notre protocole comme représenté sur la Figure 26. Nous remarquons que les temps de réponse sont distincts entre les trois courbes, la première courbe de SA montre des délais très courts, mais comme nous l'avons vu dans les graphes précédents, ce protocole ne prend pas en charge la tolérance aux défaillances ni des actionneurs ni des capteurs.

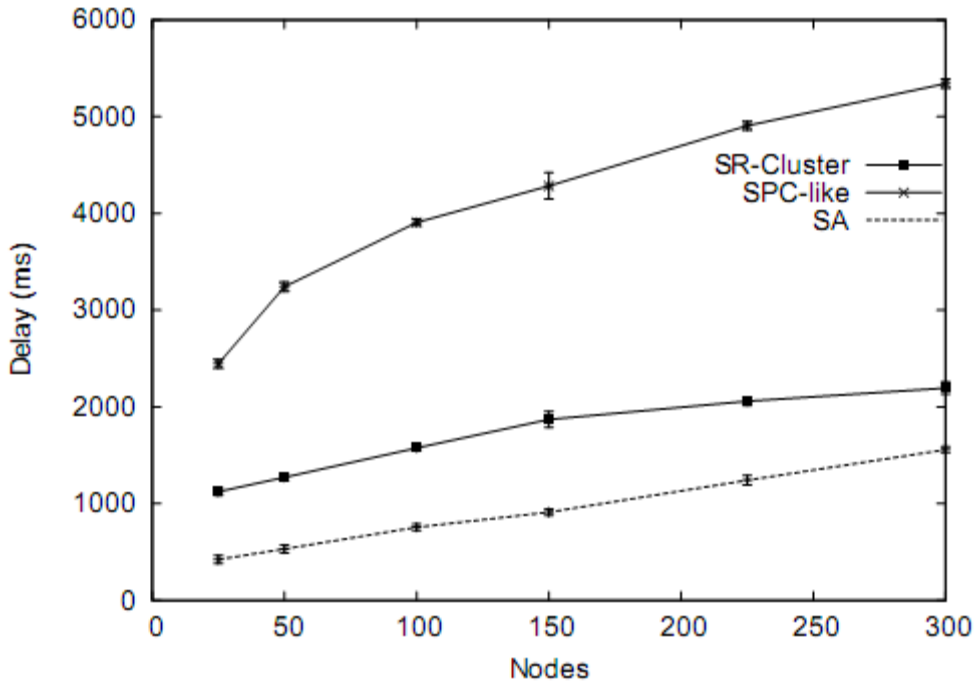

Figure 26 : Temps de latence d'exécution des taches vs Nombre de nœuds

Notre protocole SR-Cluster montre des délais plus long que SA mais nous voyons qu'il évolue presque de la même façon que la courbe SA. La différence dans les délais peut s'expliquer par le fait que dans notre protocole, pour faire face à une panne d'un CH-primaire, le CH-secondaire attend un temps limite *(WaiteForAck)* avant de réagir dans l'environnement du CH-primaire. Ce temps est nécessaire pour éviter les exécutions redondantes d'une action. Le temps WaiteForAck correspond au temps de transition du message entre le CH-primaire et le CH-secondaire. Les délais de réaction dans l'environnement peuvent être plus courts en utilisant la QoS (communication temps réel) et en diminuant la valeur de la constante WaiteForAck. La construction des CA se fait avant l'envoi de la donnée, donc elle n'influe pas sur les temps de réponse.

Pour le protocole SPC-Like nous voyons qu'il évolue de façon presque linéaire avec un décalage visible en termes de temps de réaction entre la courbe SR-Cluster et SPC-Like. Cette différence s'explique par le fait que le consensus est exécuté après l'envoi de la donnée captée, ou plutôt, après la réception de la donnée par les actionneurs backup, qui doivent la retransmettre vers l'actionneur primaire et attendre aussi un acquittement de la part de ce dernier (Coordination semi passive SPC). Dans le cas de panne de l'actionneur primaire, les actionneurs backup doivent élire un autre

actionneur backup pour devenir primaire. Pour se faire, un processus de consensus est lancé entre tous les actionneurs backup. Ce consensus prend relativement plus de temps à chaque fois que le nombre de nœuds dans le réseau augmente, mais surtout à chaque fois que le nombre d'actionneurs augmente aussi.

Dans l'exécution des deux scénarii précédant, nous avons aussi remarqué que la communication inter actionneurs dans le protocole SPC dans le but d'élire un actionneur primaire, peut générer l'envoi de beaucoup de paquets supplémentaires qui devront être traité par les nœuds capteur intermédiaires. Cette communication de groupe entre les actionneurs pour réaliser un consensus, cause une latence supplémentaire, augmente la consommation énergétique des capteurs et diminue la durée de vie du réseau, elle peut aussi poser des problèmes de déconnexion dans le réseau.

En contrepartie notre protocole ne génère pas de paquets de communication supplémentaire après l'établissement d'un CA. Il permet ainsi de minimiser la consommation énergétique des nœuds capteurs vu que l'envoi de donnée se fait de façon localisé dans CA.

7. Conclusion :

TOSSIM se trouve être un simulateur de réseau de capteurs très puisant. Il est le mieux adapté pour simuler un WSAN même s'il ne prend pas en charge les caractéristiques particulières des actionneurs. Nous avons pu utiliser TOSSIM dans sa version 2.0 pour réaliser nos simulations et les résultats obtenue ont démontré l'efficacité de notre algorithme par rapport aux deux autres protocoles qui nous avons implémenté. La simulation a aussi démontré que notre protocole garde toujours un niveau d'erreur acceptable au-delà de 88% même avec un taux de panne des actionneurs qui est égale à 50%.

La comparaison de notre protocole avec le protocole SPC-Like démontre les avantages de notre technique de clustering qui permet d'obtenir des clusters de taille égale. De cette façon, l'actionneur primaire d'un CA se trouve toujours très proche du capteur qui déclenche l'événement et peut par conséquent répondre rapidement. Contrairement au protocole SPC, qui se base sur un seul actionneur actif et qui a

démontré ses limites en termes de scalabilitée et surtouts par rapport au temps de latence.

La consommation d'énergie est aussi un point positif pour notre protocole qui une fois le clustering réalisé, il ne transmet pas de message au-delà du cluster CA et tous ces communications ce font en unicast ou en multicast vers les deux actionneurs d'un CA uniquement. Il permet ainsi une économie considérable d'énergie par rapport au protocole SPC. La QoS implémenté dans les messages DATA, est un autre moyen d'économiser l'énergie moyenne dépensée par tous les nœuds du réseau. En effet, un capteur qui juge que la nature de l'événement n'est pas à temps réelle, il peut utiliser la communication MA-LEP pour transmettre son message.

Nous avons inscrit les simulations de consommation d'énergie pour les trois protocoles comme perspective dans un proche avenir, ce qui nous permettra de démontrer l'efficacité énergétique de notre protocole qui se base sur la QoS et les chemins LEP pour minimiser la consommation moyenne du réseau.

Conclusion

La nécessité des WSANs s'est fait sentir par ses divers domaines d'application, qu'ils soient civils ou militaires, dans un environnement d'intérieur, ou pour explorer des conditions extrêmes. Elle s'est d'abord fait sentir par son enjeu économique et par les qualités de service qu'elle permet d'obtenir en précision et en temps de réponse.

Néanmoins, cette classe de réseaux émergente soulève un certain nombre de défis, telle une coordination entre des entités hétérogènes, pour aboutir à un routage à temps réel avec une consommation énergétique minimale, afin d'assurer une durée de vie plus longue au réseau. Rajouter à cela un aspect critique dans ce type de réseaux qui est la tolérance aux défaillances. Par exemple, une défaillance dans le système de détection d'incendies et d'évacuation peut engendrer des pertes humaines.

La tolérance aux défaillances est un chalenge dans les WSANs en particulier la tolérance aux pannes des actionneurs qui sont déployés avec un nombre limité et doivent parfois être mobile pour couvrir une région plus grande. Ces actionneurs en plus des pannes traditionnelles de batterie et d'interface radio, ils sont sujets à des pannes mécaniques sur l'unité d'actuation. Comme nous avons pu le voir dans ce travail les techniques de redondance et de réplication ne sont pas adapté pour toléré les pannes des actionneurs. La première exige un grand nombre d'actionneurs, et la seconde augmente la consommation d'énergie des capteurs qui font transiter la donnée dupliqué entre les différents actionneurs.

Nous avons proposé une technique nouvelle qui permet de tolérer les pannes des actionneurs tout en minimisant le nombre d'actionneurs à déployer et l'énergie consommée par les capteurs dans le réseau. Notre technique de tolérance aux défaillances est basée principalement sur un protocole de clustering rapide et locale. Il permet d'obtenir des clusters autoréparables de taille égales, chaque cluster contient deux actionneurs CHi et CHj. La gestion d'un cluster *Ci* en interne incombe au *CHi.* Dans un cas de panne d'un *CHi* le *CHj* voisin membre d'un même CA, peut le remplacé dans toute ces taches de décision et d'actuation. Que nous soyons dans un schéma capteurs actionneurs ou capteurs actionneur et équipements d'actuation le clustering fonction de la même manière.

Nous avons aussi proposé un protocole de routage qui permet deux types de communication tolérant aux défaillances, qui peuvent être utilisés suivant le domaine d'application des WSANs. Les communications à temps réel pour les applications qui ne supportent pas un temps de latence, et les communications à consommation d'énergie minimale pour les applications à prélèvements périodiques.

Nos résultats de simulation sous TOSSIM ont démontré l'efficacité de notre algorithme par rapport aux deux autres protocoles de comparaison. La simulation a aussi démontré que notre protocole garde toujours un niveau d'erreur acceptable au-delà de 88% même avec un taux de panne des actionneurs égale à 50%.

La consommation d'énergie est aussi un point positif pour notre protocole qui une fois le clustering réalisé, il ne transmet pas de message au-delà du cluster CA et tous ces communications ce font en unicast ou en multicast vers les deux actionneurs d'un CA uniquement. Au lieu d'un multicast vers tous les actionneurs comme nous l'avons vu dans le model SPC. Notre protocole permet ainsi une économie considérable d'énergie par rapport à SPC. Nous avons inscrit les simulations de consommation d'énergie pour les trois protocoles comme perspective dans un proche avenir, ce qui nous permettra de démontrer l'efficacité énergétique de notre protocole qui se base sur la QoS et les chemins LEP pour minimiser la consommation moyenne du réseau.

La perspective d'adapté notre protocole de routage à la norme IPv6, est de mise, vu que ce deniers est le successeur légitime du protocole IPv4 et qui permet d'obtenir autant d'adresse pour déployé des très grands WSANs avec comme principale avantage la possibilité de communication directe entre plusieurs WSANs sans besoin de NAT ou autres. Une implémentation de Micro IP a été implémenté en 2008 pour permettre aux capteurs d'adapté un adressage IPv6 [DAW08].

La gestion de la micro-mobilité des actionneurs dans un même cluster CA et de la macro-mobilité au de la du cluster CA est une bonne problématique en perspective surtout avec les contrainte d'énergie et de latence. L'adaptation de notre algorithme de clustering a ce type de mobilité peut engendrer moins de perte de paquets du a l'indisponibilité du nœud actionneur dans sa zone d'événement primaire.

Bibliographies

[ABT04] Adam Dunkels, Björn Grönvall, Thiemo Voigt, « Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors », 29th Annual IEEE International Conference on Local Computer Networks, Pages: 455–462, 2004.

[AkK04] Akyildiz, I. F. and Kasimoglu, I. H., "Wireless sensor and actor networks: Research challenges," Ad Hoc Networks (Elsevier), vol. 2 N:4, pp. 351–367, October 2004.

[ALM05] Th. Arampatzis, J. Lygeros, and S. Manesis, "A Survey of Applications of Wireless Sensors and Wireless Sensor Networks" Proceedings of the 13th Mediterranean Conference on Control and Automation Limassol, Cyprus, June 27-29, 2005

[ASSC02] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, A Survey on Sensor Networks, IEEE Communications Magazine, August 2002.

[BoM07] Boukerche, Azzedine; Martirosyan, Anahit "An Efficient Algorithm for Preserving Eventsapos; Temporal Relationships in Wireless Sensor Actor Networks" 2007. LCN 2007. 32nd IEEE Conference on Local Computer Networks Volume , Issue , 15-18 Oct. 2007 Page(s):771 – 780

[CCJ06] Cormac Duffy, Cormac J. Sreenan, John Herbert, Utz Roedig, « A Performance Analysis of MANTIS and TinyOS », Technical Report CS-2006-27-11, University College Cork, Ireland, November 2006.

[CEM05] C. Han, E. Kohler, M. Srivastava, R. Kumar, R. Shea, « A Dynamic Operating System for Sensor Nodes», Proceedings of the 3rd International Conference on Mobile Systems, Applications and Services (Mobisys), Page(s): 163-176, University of California, Los Angeles, June 2005.

[CGMT04] M. Conti, S. Giordano, G. Maselli, G. Turi, Cross-layering in mobile ad-hoc network design, IEEE Computer magazine, Special Issue on AdHoc Networks 37 (2) (2004) 48–51.

[ChM05] S. Chessa, and P. Maestrini, "Fault Recovery Mechanism in Single-hop Sensor Networks," Computer Communications, vol. 28, issue 17, pp. 1877-1886, 2005.

[ChS05] Y. Chen, and S.H. Son, "A Fault Tolerant Topology Control in Wireless Sensor Networks," Proc. of the ACS/IEEE 2005 International Conference on Computer Systems and Applications, 2005.

[ChS07] Chaudhari, Qasim M.; Serpedin, Erchin "A Simple Algorithm for Clock Synchronization in Wireless Sensor Networks" World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007on a Volume, Issue, 18-21 June 2007 Page(s):1 – 4

[CRO] "Crossbow MICA2 Mote Specifications." http://www.xbow.com.

[CYW07] M. Cardei, S. Yang, and J. Wu, "Algorithms for Fault-Tolerant Topology in Heterogeneous Wireless Sensor Networks," to appear in IEEE Transactions on Parallel and Distributed Systems, 2007.

[DAW08] Making Sensor Networks IPv6 Ready. Mathilde Durvy, Julien Abeillé, Patrick Wetterwald, Colin O'Flynn, Blake Leverett, Eric Gnoske, Michael Vidales, Geoff Mulligan, Nicolas Tsiftes, Niclas Finne, and Adam Dunkels. Proceedings of the 6th ACM Conference on Embedded Networked Sensor Systems (SenSys'08), Raleigh, USA, November 2008.

[DDHC04] Wenliang Du, Jing Deng, Yunghsiang S. Han, Shigang Chen and Pramod Varshney, A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge, in IEEE INFOCOM'04, March 7-11, 2004, Hongkong.

[Dem04] M. Demirbas, "Scalable Design of Fault-Tolerance for Wireless Sensor Networks," PhD Dissertation, The Ohio State University, 2004.

[DoP03] Donggang Liu and Peng Ning, Efficient Distribution of Key Chain Commitments for Broadcast Authentication in Distributed Sensor Networks, Proceedings of the 10th Annual Network and Distributed System Security Symposium, pages 263 - 276, San Diego, California. February 2003.

[DPH06] David Gay, Philip Levis, « TinyOS Programming », Livre, ISBN: 0521896061, Presse de l'université de Cambridge, 28 Juin 2006.

[FWW06] Lin Fan; Huanzhao Wang; Hai Wang "A solution of multi-target tracking based on FCM algorithm in WSN". 2006. Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops Volume , Issue , 13-17 March 2006

[HaZ07] Hu Haifeng; Yang Zhen "Mobile-Agent-Based Information-Driven Multiresolution Algorithm for target tracking in Wireless Sensor Networks" Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference on Volume 1, Issue , July 30 2007-Aug. 1 2007 Page(s):521 – 525

[HJK08] H. Alatrista, J. Mathieu, K. Gouaïch S. Aliaga, « Implémentation de protocoles sur une plateforme de réseaux de capteurs sans fil », TER master 1 informatique, Université de Montpellier II, 29 Avril 2008.

[HSS07] F. Hu, W. Siddiqui, K. Sanka: Scalable security in wireless sensor and actuator networks (WSANs): integration re-keying with routing, Computer Networks: The International Journal of Computer and Telecommunications Networking, January 2007, vol 51, pp. 285 - 308

[JGK07] Jorgic, Milenko Goel, Nishith Kalaichevan, Kalai Nayak, Amiya Stojmenovic, Ivan : Localized detection of k-connectivity in wireless ad hoc, actuator and sensor networks, 2007. Proceedings of 16th International Conference on Computer Communications and Networks Aug. 2007 p. 33-38.

[KPS04] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentelli, "Fault Tolerance in Wireless Sensor Networks," Book chapter in Handbook of Sensor Networks, I. Mahgoub and M. Ilyas (eds.), CRC press, Section VIII, no. 36, 2004.

[Krl04] B. Krishnamachari, and S. Iyengar. "Distributed Bayesian Algorithms for Fault-Tolerant Event Region Detection in Wireless Sensor Networks," IEEE Transactions on Computers, vol. 53, no. 3, pp. 241-250, 2004.

[LNS07] Hai Liu, Amiya Nayak, Ivan Stojmenovic, "Fault Tolerant Algorithms/ Protocols in Wireless Sensor Networks", www.site.uottawa.ca/~hailiu/publications/FaultTolerance07.pdf.

[MiH05] Daniel Minder, Jorg Hahner "à comparison of the architecture of the network simulators NS-2 and TOSSIM. In Seminars Performance Simulation of Algorithms and Protocols, Januar 2005

[MPA06] Tommaso Melodia, Dario Pompili, Ian F. Akyildiz "A Communication Architecture for Mobile Wireless Sensor and Actor etworks" sensor ad-hoc communication and networks (SECON'06) sep 2006, Atlanta GA, pp.109-118.

[MWI06] Mathieu Badnet, Nicolas Belloir «Réseaux de capteurs : Mise en place d'une plateforme de test et d'expérimentation », Master Technologie de l'Internet 1ère année, France, 2005/2006.

[OHE07] Keiji Ozaki, Naohiro Hayashibara, Tomoya Enokido, Makoto Takizawa "Fault-Tolerant Semi-Passive Coordination Protocol for a Multi-Actuator/Multi-Sensor (MAMS) Model" The Second International Conference on Availability, Reliability and Security IEEE Computer Society Washington, DC, USA, pages 506-516. Year 2007.

[OWI06] Keiji Ozaki, Kenichi Watanabe, Satoshi Itaya, Naohiro Hayashibara : "TomoyaEnokidoAFault-TolerantModelforWirelessSensor-ActorSystem" Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA'06). Year 2006.

[ShY07] Shih-Lin Wu, Yu-Chee Tseng, "WIRELESS AD HOC NETWORKING: Personal-Area, Local-Area, and the Sensory-Area Net", Auerbach Publications, 2007.

[SYL07] Sylvie Tixier, « TinyOS », Mini rapport, LIF12, Université Lyon 1, 6 Décembre 2007.

[TPS05] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong, "A macroscope in the redwoods," In SenSys'05: Proceedings of the 3rd international conference on Embedded networked sensor systems, New York, NY, USA, 2005.

[VZS06] Ramanuja Vedantham, Zhenyun Zhuang and Raghupathy Sivakumar "Mutual Exclusion in Wireless Sensor and Actor Networks" 2006

[YAC08] Yacine Challal, « Réseaux de Capteurs Sans Fils », Cours, Systèmes Intelligents pour le Transport, Université de Technologie de Compiègne, France, 17 Novembre 2008.