Vers une application d'un réseau de capteurs sans fil dans la problématique des feux de forêt : Modélisation, Simulation et Plate-forme expérimentale

Thierry Antoine-Santoni

UNIVERSITÉ DE CORSE – PASQUALE PAOLI

U.F.R. SCIENCES ET TECHNIQUES

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE CORSE

ÉCOLE DOCTORALE ENVIRONNEMENT ET SOCIÉTÉ

Discipline : Sciences pour l'Environnement

Spécialité : Informatique

présentée par

Thierry Antoine-Santoni

Vers une application d'un réseau de capteurs sans fil dans la problématique des feux de forêt : Modélisation, Simulation et Plate-forme expérimentale

---

sous la direction du Professeur

Jean-François SANTUCCI

soutenue publiquement le 14 décembre 2007 devant le jury composé de :

Rapporteurs : M. Jean-Jacques CHABRIER, *Professeur*, *Université de Bourgogne*

M. Denis FLOUTIER, *Professeur*, *École Polytechnique Universitaire de Savoie*

Examinateurs : M. Gérard-André CAPOLINO, *Professeur*, *Université de Picardie*

M. David HILL, *Professeur*, *Université Blaise Pascal*

M. Paul-Antoine BISGAMBIGLIA, *Professeur*, *Université de Corse*

M. Jean-François SANTUCCI, *Professeur*, *Université de Corse*

Mme Emmanuelle De GENTILI, *Maître de Conférence*, *Université de Corse*

Mme Bernadette COSTA, *Maître de Conférence*, *Université de Corse*

UNIVERSITÉ DE CORSE – PASQUALE PAOLI

U.F.R. SCIENCES ET TECHNIQUES

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE CORSE

ÉCOLE DOCTORALE ENVIRONNEMENT ET SOCIÉTÉ

Discipline : Sciences pour l'Environnement

Spécialité : Informatique

présentée par

Thierry Antoine-Santoni

Vers une application d'un réseau de capteurs sans fil dans la problématique des feux de forêt : Modélisation, Simulation et Plate-forme expérimentale

---

sous la direction du Professeur

Jean-François SANTUCCI

soutenue publiquement le 14 décembre 2007 devant le jury composé de :

Rapporteurs : M. Jean-Jacques CHABRIER, *Professeur*, *Université de Bourgogne*

M. Denis FLOUTIER, *Professeur*, *École Polytechnique Universitaire de Savoie*

Examinateurs : M. Gérard-André CAPOLINO, *Professeur*, *Université de Picardie*

M. David HILL, *Professeur*, *Université Blaise Pascal*

M. Paul-Antoine BISGAMBIGLIA, *Professeur*, *Université de Corse*

M. Jean-François SANTUCCI, *Professeur*, *Université de Corse*

Mme Emmanuelle De GENTILI, *Maître de Conférence*, *Université de Corse*

Mme Bernadette COSTA, *Maître de Conférence*, *Université de Corse*

A minnanna...

Remerciements

Cette thèse s'est déroulée au sein de l'équipe Modélisation Informatique du laboratoire Systèmes Physiques pour l'Environnement (Unité Mixte de Recherche CNRS 6134) de l'Université de Corse. Ce travail a été réalisé grâce au soutien financier de la Collectivité Territoriale de Corse. Que ces deux institutions trouvent ici le témoignage de ma reconnaissance.

Je tiens tout particulièrement à exprimer ma plus profonde gratitude à Monsieur Jean-François SANTUCCI, Professeur à l'Université de Corse qui m'a accueilli dans son laboratoire, pour avoir dirigé ces travaux et m'avoir soutenu dans cette étude.

Des remerciements trés spéciaux à Madame Emmanuelle De GENTILI et Madame Bernadette COSTA, Maîtres de Conférence à l'Université de Corse, pour leur disponibilité, leurs conseils, l'aide scientifique et morale qui ont été indispensables à la concrétisation de cette recherche.

À Monsieur Jean-Jacques CHABRIER, Professeur à l'Université de Bourgogne, et à Monsieur Denis FLOUTIER, Professeur à l'Ecole Polytechnique Universitaire de Savoie, j'adresse ma plus respectueuse reconnaissance pour l'intérêt qu'ils ont porté à ce travail en acceptant d'en être les rapporteurs dans ce jury.

Que Monsieur Gérard-André CAPOLINO, Professeur à l'Université de Picardie, et Monsieur David HILL, Professeur à l'Université Blaise Pascal qui me font l'honneur de participer à ce jury, trouvent ici l'expression de ma profonde gratitude.

Je voudrais également exprimer toute ma reconnaissance à Monsieur Paul-Antoine BISGAMBIGLIA pour avoir accepté de participer à ce jury.

Je remercie tous les chercheurs et personnels de l'Université de Corse, avec une pensée spéciale pour Christophe, Marie-Laure, Laurent, Céline, François, l'équipe feu, en particulier Xavier et Fredéric, pour leur grande gentillesse et leur contribution à l'élaboration de ce travail.

Ma gratitude va à mes parents et ma famille qui ont toujours été présents dans les moments difficiles durant toutes ces années d'études. J'exprime également ma gratitude à Antoinette pour son soutien et sa patience à supporter les contraintes qui découlaient de ce travail. Finalement, un grand merci va à tous mes amis et à mes collègues Sapeurs-Pompiers à COZZANO pour leur amitié et leur soutien sans faille.

Table des matières

Introduction générale

"Les deux mots les plus brefs et les plus anciens, oui et non, sont ceux qui exigent le plus de réflexion."

Phytagore

La lutte contre les feux de forêt est un enjeu primordial en ce début de XXI^{ime} siècle. Au delà du processus affirmé du rechauffement climatique, la survie de notre système environnemental est évidemment le point de liaison de nombreuses recherches. En 2005, la Péninsule Ibérique, en proie à de violents incendies, voit le Portugal ravagé par les feux détruisant entre 180 000 et 220 000 hectares de végétation. Dans le même temps, l'Espagne n'était pas épargnée. Le bilan 2005 des feux de forêt en France est également dramatique. Le Conseil des ministres a dressé le bilan de la campagne de lutte contre les incendies en 2005 avec 17 000 hectares de végétation détruits dans les départements méditerranéens. La problématique feu est l'un des axes de recherches de l'Université de Corse, en particulier celle du laboratoire des Sciences Physiques pour l'Environnement (SPE), de l'Unité Mixte de Recherche CNRS 6134. Vu les catastrophes de cette dernière décénie, que ce soit au Portugal, aux Etats-Unis mais également en Corse avec la désastreuse saison 2003, les services de secours avouent la nécessité de disposer de moyens efficaces de prévention, de détection et de lutte contre les feux de forêt. Nous voulons nous inscrire dans cette optique en fournissant une étude sur un nouvel outil de surveillance des feux de

forêt : les réseaux de capteurs sans fil. La convergence de trois technologies majeures a contribué à l'éclosion de cette nouvelle génération de réseau [Hac, 2003]. Ces trois domaines sont les circuits digitaux, les communications sans fil et les micro-systèmes électro-mécaniques. Les avancées dans le dévelopement des supports matériels et de la définition des architectures ont permis la réalisation d'entités de petite taille, consommant peu d'énergie et de faible coût. Les noeuds de ce type de réseaux consistent en un grand nombre de micro-capteurs capables de récolter et de transmettre des données environnementales d'une manière autonome. Ils sont dispersés à travers une zone géographique définissant le terrain d'intérêt pour le phénomène capté. Les données captées sont acheminées grâce à un routage multi-saut, routage de proche en proche, à un noeud considéré comme un "point de collecte", appelé noeud puit ou station de base. Les données collectées sont ensuite analysées par un utilisateur.

Cependant pour aider efficacement les services de secours, l'utilisation d'un réseau de capteurs sans fil doit être capable de répondre à trois objectifs :

1. prévoir les conditions favorables à l'éclosion d'un feu,
2. détecter la naissance d'un feu
3. et permettre le suivi de cet incendie.

La détection est un facteur déterminant dans la lutte contre les feux de forêt. La rapidité d'action, des services de lutte contre les incendies, sur un phénomène naissant augmente l'efficacité des actions d'extinction entreprises. Selon [Lafarge, 2006], le manque de précision lors de l'alerte est considérée comme l'un des facteurs augmentant les délais lors des interventions. La précision et la localisation de la fumée suspecte signalée dépend fortement de la source de la première alerte. Seules les alertes données par les vigies et les patrouilles permettent un délai d'intervention de moins de 15 minutes. Selon des informations de la base de données *Prométhée*[1] [Prométhée, 1973], environ trois quarts des alertes ont été données par la population alors qu'un 5^{ime} des alertes proviennent des dispositifs de sécurité. Il apparait dans ce rapport que les patrouilles terrestres et aériennes ne détectent que 1% des feux de forêts. Selon l'étude

[1] *Prométhée* est une base de données sur les incendies de forêts de la région méditerranéenne. Conçue et lancée en 1973, cette opération couvre 15 départements du Sud-Est.

[Lafarge, 2006], l'augmentation de vigies et de points de guets diminuent de façon conséquente les surfaces brulées. Cette diminution est due aux réductions des délais d'intervention, conséquences directes de la qualité des informations fournies au secours. Les réseaux de capteurs remplissent les qualités techniques pour fournir des données environnementales précises. Au regard de ce constat, il apparaît essentiel de pouvoir fournir un outil précis de détection et de localisation d'un feu de forêt. Pour augmenter la capacité de réaction des secours face à un incendie, l'utilisation d'un réseau de capteurs sans fil est réellement intéressante.

Une notion est importante dans la compréhension de l'architecture des réseaux de capteurs sans fils : la communication. Dans les réseaux sans infrastructures, dont les noeuds communiquent par lien radio, et qui doivent s'auto-organiser, chaque noeud peut servir de routeur à ses voisins, et donc retransmettre un message reçu. Si la couverture du réseau est suffisante (c'est-à-dire qu'il existe assez de noeuds pour un espace donné), chacun peut joindre un autre, soit s'il est à portée radio, soit en utilisant des noeuds situés entre eux-deux pour relayer leurs messages.

Pour espérer une communication longue distance, le réseau va utiliser un mécanisme essentiel nommé multi-sauts (*multihop*). Le phénomène de communication multi-sauts signifie que les éléments du réseau doivent adopter une stratégie de transmission de l'information de proche en proche pour organiser l'acheminememnt de données entre deux points comme illustré sur la Figure 1.

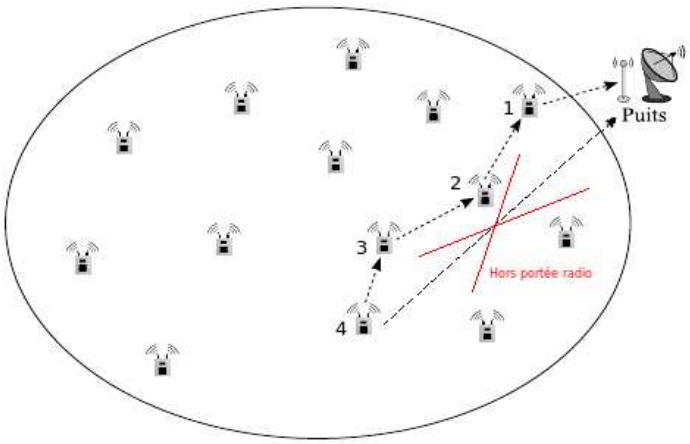


FIG. 1*: Mécanisme de communication multi-saults*

Si le Noeud 4 désire transmettre au puit, il sera dans l'impossibilité de le faire directement, limité par la portée de la radio. Les réseaux de capteurs sans fil permettent d'utiliser tout ou une partie des terminaux raccordés comme des relais actifs de communication. Cette architecture a comme avantage de se passer de point d'accès et sourtout d'être flexible et dynamique. Tous les noeuds entre eux sont capables de s'échanger des informations, soit directement, soit indirectement par l'intermédiaire des autres noeuds du réseau.

L'originalité de notre approche est de fournir une étude sur un outil de surveillance environnementale dans la lutte contre les feux de forêt en identifiant les contraintes propres au phénomène. Au regard de ces différentes contraintes de communication, de robustesse du réseau face à un phénomène destructeur, il semble pertinent de fournir des réponses concrètes dans le cas d'une application dans un futur proche.

Notre objectif principal est de fournir une analyse sur l'application des réseaux de capteurs sans fil dans la problématique de feux de forêt. Pour cela nous allons orienter nos travaux selon deux axes :

1. la modélisation et la simulation d'un réseau de capteurs pour étudier le comportement d'un tel système face un phénomène environnemental destructeur ; en effet les concepts de modélisation et de simulation s'avèrent évidents vis à vis de notre sujet d'étude ; il serait paradoxal de déployer un tel système sans connaître au préalable les comportements fonctionnels du réseau dans le cadre d'un feu de forêt.
2. l'étude d'un réseau en conditions réelles de feu de forêt sur une plate-forme expérimentale et l'évaluation de ses capacités de détection et de suivi.

Ces objectifs doivent nous permettre de dire si **oui** *ou* **non***, les réseaux de capteurs sans fil peuvent permettre la prévison, la détection et le suivi d'un feu de forêt.*

Ce mémoire s'articule autour de six chapitres. Le premier chapitre énonce la problématique de notre étude en détaillant les processus de prévision, détection, et suivi d'un feu de forêt à l'aide d'un réseau de capteurs sans fil. Dans le second chapitre nous proposons une description

du contexte d'étude présentant tout d'abord la technologie des réseaux de capteurs sans fil et nous décrivons ensuite les principaux outils permerttant d'étudier le domaine. Nous soulignons, en particulier, la nécessité de diriger vers la conception d'un nouvel outil de modélisation et de simulation dans le cadre de notre étude. Le troisième chapitre aborde les concepts de modélisation et de simulation avec une présentation du formalisme DEVS, les fondements qui vont nous conduire à la modélisation d'un capteur sans fil. Notre approche de modélisation est décrite dans le quatrième chapitre par la description d'un modèle de capteur à l'aide du formalisme DEVS et la définition d'un nouvel algorithme de routage. Le cinquième chapitre propose l'implémentation d'un nouvel outil **DEVS-WSN (DEVS-Wireless Sensor Network)** pour la simulation d'un réseau de capteurs sans fil fondé sur l'approche de modélisation informatique précédente. L'outil que nous avons développé permet de fournir les résultats de simulation de trois protocoles de routage et d'étudier les stratégies déterministes de déploiement d'un réseau de capteurs dans le cadre d'un feu de forêt. Nous complétons notre étude dans le sixième chapitre, par une expérience réalisée à l'aide d'un réseau de capteurs sans fil sur une plate-forme expérimentale "feu de forêt". Enfin nous achevons ce mémoire par une conclusion, au sein de laquelle, nous proposons un ensemble de perspectives à donner à nos travaux de recherche.

CHAPITRE 1

Problématique

"Chaque étincelle est à elle seule tout l'incendie ; elle le porte, l'augmente, le diffuse."

Extrait d' Hypathie ou la fin des dieux, *Jean Marcel*

Comme beaucoup de technologies, les applications militaires [Akyildiz et al., 2002c] ont conduit à l'émergence de capteurs sans fil. Durant la guerre froide, le sytème de surveillance sonore (SOSUS) de l'armée américaine [Chong et Kumar, 2003], système constitué de capteurs immergés, a été déployé stratégiquement sur le fond des océans pour détecter les mouvements des sous-marins soviétiques. Cette première utilisation a permis de lancer un nouvel axe de recherche. Ce système est actuellement utilisé par la *National Geographic and Atmospheric Administration* (*NOAA*) pour surveiller des mouvements dans les océans liés à l'activité terrestre ou animale. La recherche moderne sur les réseaux de capteurs a commencé dans les années 80 avec le programme sur les réseaux distribués de capteurs (*Distributed Sensor N*etwork *:DSN*) lancé par le département recherche du ministère de la défense américaine (*DARPA*). Durant le workshop *DSN* de 1978 [DSN, 1978], les composants technologiques applicables au *DSN* ont été identifiés et sont les suivants : les senseurs ou capteurs, les outils de communication, les techniques et algorithmes

de traitement des données et les suites de logiciels distribués. Les chercheurs de l'Université de Carnegie Mellon (CMU) et de l'Institut de Technologie du Massachussets (MIT) ont poursuivi cet effort en travaillant sur des systèmes d'exploitation ou sur des techniques de traitement du signal. Les premières plate-formes d'essai avaient une particularité : les éléments des réseaux se présentaient sous la forme d'entités mobiles de communication avec d'imposantes unités de réception et de traitement du signal. Bien que les chercheurs aient déjà pensé dans les années 80 à des structures de petites tailles, les différentes composantes technologiques n'étaient pas prêtes pour accéder à des éléments réduits en taille. Les avancées récentes qui ont conduit à augmenter la puissance de calcul et de communication ont causé un décalage significatif dans la recherche sur les réseaux de capteurs sans fil et permis de se rapprocher de la vision originale des chercheurs [Chong et Kumar, 2003]. L'apparition de petites sondes performantes et peu coûteuses basées sur la combinaison des technologies des Micro Systèmes ElectroMécanique *(MEMS*), la gestion de réseau sans fil, et le prix réduit des processeurs de basse puissance permettent à présent le déploiement des réseaux ad hoc sans fil pour différentes applications. En effet, les réseaux de capteurs actuels peuvent exploiter des technologies et des fonctionnalités inimaginables il y a encore 20 ans. C'est au début des années 1990 que sont posées les bases de la problématique du regroupement de plusieurs microcapteurs sur une même puce par K.D. Wise [Wise et Najafi, 1991]. L'étude s'est d'abord portée sur la compatibilité des procédés technologiques de fabrication des parties capteurs avec le traitement du signal. Quelques années plus tard, dans les années 2000, le lancement du projet "*Smart Dust*" par K. Pister [Warneke et al., 2001] tente de réaliser un regroupement de microcapteurs sur une même puce. L'objectif est d'intégrer un système de traitement du signal et d'y ajouter un moyen de communication. Les thèmes abordés par cette nouvelle problématique sont de nature différente, ils concernent principalement, du point de vue informatique, la gestion de l'énergie que consomme le système lors de son activité de calcul [Hill et al., 2000b], l'intégration variée de microcapteurs et les moyens de communication et de transmission de l'information. Les nombreuses recherches sur le plan international ont conduit, dans ces trois domaines, à l'obtention d'une première génération de systèmes avancés, développés sur une même base architecturale comme le décrit [Maurice, 2005] : capteur environnemental, traitement du signal, mémoire

et communication.

Les avancées technologiques des systèmes de communication sans fil et de la miniaturisation des composés électroniques, que ce soit au niveau des processeurs ou des capteurs, ont conduit à l'émergence d'un nouveau type de surveillance et de contrôle environnemental. A l'origine les réseaux de capteurs ont une application militaire notamment pour la recherche d'ennemis sur le terrain, de mouvements de troupes, d'environnements dangereux. La lutte contre les feux de forêt peut s'apparenter à cette problématique en identifiant trois axes majeurs dans l'utilisation des réseaux de capteurs [Antoine-Santoni et al., 2006] :

1. la prévision ou l'analyse d'un environnement propice à la naissance d'un feu de forêt,
2. la détection ou la mise en place d'une sentinelle capable de signaler des comportements environnementaux anormaux,
3. la lutte ou les moyens de suivre l'évolution d'un phénomène dans le temps et dans l'espace.

Nous proposons de faire un point sur ces troix axes en soulignant les domaines qui doivent être éclairés par notre recherche.

1.1 La prévision

Nous savons que des conditions particulières peuvent être propices à l'éclosion d'un feu de forêt. Les causes d'un sinistre sont parfois difficiles à établir avec certitude. Pour les déterminer, des enquêtes de terrain sont menées après chaque incendie. La base de données *Prométhée* permet une approche globale du phénomène en région méditerranéenne : 65 % des causes de feux y ont été identifiées, contre 35 % dans les autres départements soumis aux feux de forêt. Parmi ces causes, on distingue les facteurs anthropiques, liés aux activités humaines et les facteurs naturels de déclenchement, liés aux conditions du milieu :

– l'influence des facteurs naturels : les conditions météorologiques et les caractéristiques de la végétation conditionnent le développement des incendies, les premières pouvant avoir une influence non négligeable sur les secondes. Dans certaines situations (forts vents par exemple), la topographie du site peut également favoriser le développement des incendies ;

- les conditions météorologiques : les périodes de sécheresse et les épisodes de vents forts, sont favorables à l'éclosion des incendies. Ainsi le vent accélère le dessèchement des sols et des végétaux et augmente les risques de mises à feu, par la dispersion d'éléments incandescents par exemple ;
- les caractéristiques de la végétation : la prédisposition de la végétation aux incendies est souvent liée à sa teneur en eau, elle-même déterminée par les conditions météorologiques ;
- les conditions orographiques : dans une zone sans relief, un départ de feu est facilement soumis à l'accélération du vent. En zone de relief irrégulier, la progression du feu est accélérée dans les "montées" et ralentie dans les "descentes".

Aux vues de ces caractéristiques, il serait intéressant de disposer d'un outil capable de fournir des données terrains en temps réel. Imaginons, au delà des stations météo classiques, un outil disposé sur le terrain qui puisse fournir des données telles que la température, le taux d'humidité, la présence de vent sur une zone relativement inaccessible. L'objectif d'un tel outil serait de fournir des données sur de possibles zones à risques, identifiées par une température élevée ou un taux d'humidité relativement bas, comme montré sur la Figure 1.1.

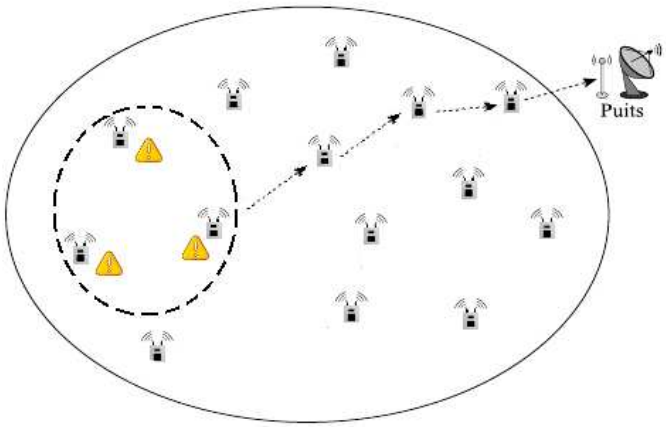


FIG. 1.1*: Visualisation de zones à risques*

Les réseaux de capteurs sans fil ont la capacité de fournir ce type de données et présentent une cartographie décrivant les zones à forts risques d'éclosion d'un feu de forêt. Cela pourrait fournir

une aide précieuse dans les Dispositifs Avancés (D.A) mis en place par les sapeurs-pompiers dans le cadre des “campagnes feu de forêt”.

1.2 La détection

La détection est une caractérisque essentielle dans la lutte contre un feu de forêt. En effet, la vitesse de réaction des secours et le temps de détection d’un incendie sont deux paramètres qui peuvent déterminer l’ampleur d’un sinistre. Il est évident que les conditions météo (fort vent), la végétation ou les données topographiques (forte pente), jouent sur l’étendue de la surface en cours de combustion. Cependant une rapidité d’action des pompiers sur un feu naissant peut augmenter l’impact des secours sur ce dernier. Un outil permettant de détecter rapidement un feu serait très intéressant afin de diagnostiquer plus rapidement le déclenchement d’un feu. En effet selon le constat effectué par l’Ecole Nationale du Génie Rural et des Eaux et des Forêts [Lafarge, 2006], le manque de précision lors de l’alerte est un des facteurs augmentant les délais lors des interventions. Il est noté que le délai de l’intervention dépend de la précision et de la localisation de la fumée suspecte signalée et que celui-ci est donc influencé par la source de la première alerte. Seules les alertes données par les vigies et les patrouilles permettent un délai d’intervention de moins de quinze minutes. Selon l’analyse faite à partir de la base de données Prométhée entre 1973 et 2005, 71% des alertes ont été données par la population et 18% des alertes proviennent des dispositifs de sécurité (patrouilles, vigies ou guet). Il apparaît dans ce rapport que les patrouilles terrestres et aériennes ne détectent que 1% des feux de forêts. Ceci est expliqué par le fait que ces patrouilles doivent être proches de ces fumées pour pouvoir les détecter et les localiser avec précision.

La notion de délai d’intervention et de précision dans la détection apparaissent comme des paramètres essentiels sur la future action des secours. Il serait utile de pouvoir disposer d’un outil qui détecte en temps réel la présence ou non d’un feu de forêt en le localisant avec précision. Cet aspect peut être résolu avec les réseaux de capteurs sans fil. En effet, en détectant une élévation rapide de température dans une zone localisée par GPS, les réseaux de capteurs sans fil peuvent au

moyen de leur réseau maillé, comme illustré sur la Figure 1.2, transmettre l'information et lancer l'intervention des secours.

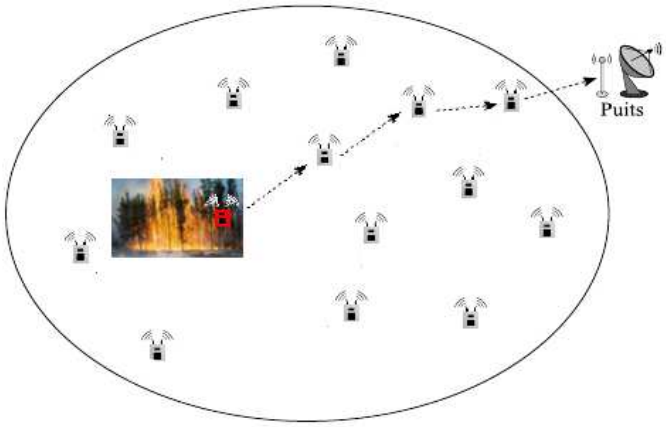


FIG. 1.2*: Détection d'un feu naissant*

Mais il faut également étudier les caractéristiques techniques telles que le moyen d'acheminer les données détectées pour espérer une transmission sans faille de l'information.

1.3 La lutte

Un moyen de cartographier une zone à risque couplé à un outil de détection en temps réel serait incomplet sans la possibilité de suivre l'évolution d'un phénomène. Il serait impensable de déployer un dispositif de détection sans pouvoir l'utiliser pour suivre le phénomène dans le temps et dans l'espace, en particulier dans le cas de feux de forêt. Nous savons qu'un réseau de capteurs sans fil s'organise sous la forme d'une structure maillée où chaque élément peut communiquer des informations avec ses voisins directs, dans le but de les transmettre à une station réceptrice. Dans la problématique des feux de forêt, le réseau doit nous permettre de visualiser un phénomène important de la manière la plus précise possible sur une surface assez grande, comme illustré sur la Figure 1.3.

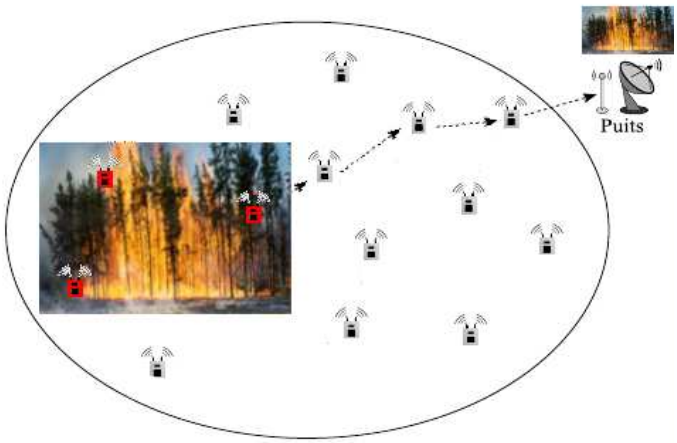


FIG. 1.3*: Suivi de l'évolution d'un feu de forêt*

Cependant, il nous semble important d'étudier le comportement d'un tel réseau face à un phénomène destructeur tel que le feu de forêt. En effet, il est nécessaire d'étudier l'évolution de cette structure dans le cas d'une importante destruction d'unités de communication. Nous ne savons pas pour l'instant comment va réagir le réseau pour transmettre les informations si une partie de ce même réseau est défaillante. A partir de ce constat, le concept de déploiement du réseau semble émerger et une étude sur la stratégie de déploiement semble nécessaire pour disposer d'un réseau efficace en toutes circonstances.

1.4 Conclusion

La prévision, la détection et le suivi d'un feu sont les axes de notre problématique. La prévision permettra de déterminer si une zone est propice à l'éclosion d'un feu. La collecte de données environnementales doit nous informer de conditions environnementales critiques (température élevée par exemple). La détection consistera à identifier, lors de la reception des données, une valeur critique (dépassement d'un seuil de température) signifiant la naissance d'un feu. Le suivi permettra d'évaluer la position d'un feu et de pouvoir prévoir son évolution. La solution que représentent les réseaux de capteurs sans fil est intéressante. Cependant cette dernière représente un nouveau type

de réseau, soumis à de nombreuses contraintes physiques et matérielles que nous nous proposons de détailler dans la partie suivante.

CHAPITRE 2

Contexte d'étude

"While the last 50 years have been dominated by a march to ever more complex computers, the next few decades will see the rise of simple sensors—by the billions."

extrait de *Business Week 1999*

Nous proposons dans le cadre de notre étude, une présentation du domaine des réseaux de capteurs sans fil. Pour offrir une meilleure lisibilité, il est nécessaire de préciser les contours de ce domaine d'étude en définissant les notions essentielles.

Un réseau *ad-hoc* est formé par un ensemble d'entités, qui s'organisent seuls et de manière totalement décentralisée, dans le but de former un réseau autonome et dynamique ne reposant sur aucune infrastructure filaire. Le fonctionnement du réseau est totalement distribué ; il n'y a pas d'élément structurant hiérarchiquement le réseau ou permettant de transmettre les trames d'un éléments à une autre. Chaque entité exécute des composantes et utilise un intergiciel, qui s'occupe d'activer les composantes et de coordonner leurs activités de tel sorte qu'un utilisateur perçoive le système comme un unique système intégré. La définition la plus courante d'un réseau *ad-hoc* est en fait "*une collection de noeuds sans-fils qui peuvent s'autoconfigurer pour former un réseau sans l'aide d'aucune infrastructure*" comme le propose [Chelius, 2004]. Les réseaux de

capteurs sans fil sont vus comme un domaine d'application des réseaux *ad-hoc*. Ces réseaux ne contiennent donc, par définition, aucune entité centrale et les connexions de ce réseau particulier permettent à chaque entité d'utiliser differentes voies pour atteindre la destination voulue. Dans le cas d'une rupture de liaison, un noeud peut continuer à communiquer avec un autre noeud en utilisant d'autres voies de transmissions. Cette architecture est l'une des clés du succès d'Internet comme le précise [Chelius, 2004], car elle donne un système robuste et une forme d'égalité entre participants. D'ailleurs, les applications les plus populaires sur Internet sont tous les programmes appelés communément P2P, (d'égal-à-égal ou peer-to-peer) qui permettent à chacun de communiquer et d'échanger des informations sans le recours d'instances dirigeantes pour la recherche d'informations.

Les réseaux de capteurs s'intégrent donc très largement dans le concept des réseaux *ad-hoc* grâce à leur nature décentralisée et leur utilisation des ondes radio comme support de communication. De même, les capteurs sont aptes à découvrir leur environnement et à s'auto-organiser pour effectuer les tâches de surveillance et de détection. Pour réaliser ces tâches, un capteur sans fil utilise des outils de communication, de traitement de l'information et de surveillance environnementale. Il est possible de définir une structure matériellle générale, commune à tous les capteurs, que nous nous proposons de détailler.

2.1 Description d'un capteur sans fil

Les réseaux de capteurs sans fil représentent la convergence des avancées technologiques de ces trente dernières années :

- les capteurs : la miniaturisation de strucures électroniques MEMS (*Micro Electro Mechanical Systems*) ;
- le réseau sans fil : nouveaux systèmes de communications sans fil tel que le WIFI ou Bluetooth basé sur les normes 802.11 et 802.15.4 ;
- la puissance de calcul des systèmes d'exploitation embarqués petits et peu coûteux en énergie et en ressources.

2.1.1 La définition d'un capteur par ses composants

Un noeud possède donc des capacités de calcul, de communication et de détection environnementale. Pour représenter les composants élémentaires d'un capteur, nous choisissons une représentation simplifiée que nous fournit [Blumenthal et al., 2003].

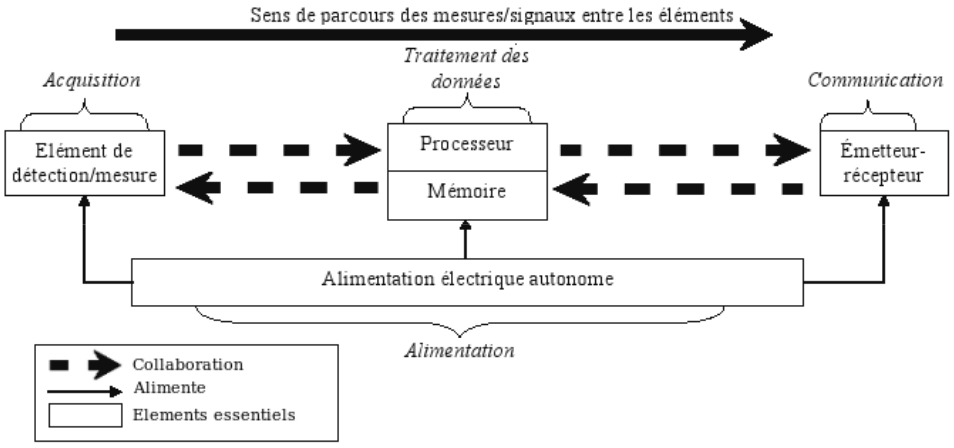


FIG. 2.1*: Structure d'un noeud*

Sur la Figure 2.1, les principaux éléments de la structure d'un noeud sont représentés. Il est possible de distinguer à partir de cette représention cinq principaux acteurs dans la composition d'un capteur sans fil :

1. l'unité d'acquisition et de détection environnementale formée des différents capteurs mais aussi d'éléments tels qu'un module GPS. Cette structure est généralement désignée par le terme panneau de sondes ou *"Sensorboard*". La principale activité de cette unité est de capter et de mesurer les données physiques de l'environnement dans la zone de détection ;
2. l'outil de traitement des données composé par le processeur et implicitement le système d'exploitation embarqué. Ils jouent un rôle majeur dans la gestion des différents éléments du capteur mais également dans la collaboration des capteurs dédiés à une application ;
3. l'unité de stockage (mémoire) intrinsèque au capteur a pour rôle de stocker ou d'agréger des informations. En tant que RAM ou que ROM, elle a un rôle essentiel dans la gestion des données mais également dans la configuration architecturale et applicative du capteur ;

4. l'unité d'émission/réception qui est constituée par l'association d'une antenne et de standards de la technologie sans fil. Plusieurs techniques existent pour permettre les échanges d'informations entre les capteurs : communication optique, infra-rouge et ondes radios. Dans le cas des ondes radios, qui est la technique la plus utilisée, il est possible de trouver plusieurs méthodes : la technologie WIFI, bluethooth et le standard *Zigbee* qui sera présenté dans la partie suivante ;
5. La source d'énergie doit alimenter les différents composants du capteur. C'est un point sensible dans l'activité d'un noeud. La source d'énergie peut revêtir plusieurs formes : piles alcaline rechargeables ou non, batterie lithium ou énergie solaire[Chong et Kumar, 1999].

Nous venons de voir les composants d'un capteur sans fil. Ces capteurs sont soumis à des contraintes de taille, de communication et surtout de consommation énergétique. Pour réduire ces contraintes et mutualiser les recherches dans ce domaine, un standard a été développé : le standard *Zigbee.* Basé sur la norme IEEE 802.15.4, ce standard propose un protocole de haut niveau que nous proposons de détailler dans la partie suivante.

2.1.2 Pile protocolaire *Zigbee* et norme IEEE 802.15.4

Les réseaux de capteurs sans fil sont foncièrement différents des réseaux traditionnels ou filaires. Les contraintes physiques, matérielles et énergétiques font de ce système, un type particulier de réseau. Les recherches intenses pour réduire ces contraintes se sont fédérées autour d'un standard : *ZigBe*e. *Zigbee* est un protocole de haut niveau permettant la communication de petites radios, à consommation réduite, basé sur le standard IEEE 802.15.4 pour les réseaux à dimension personnelle (Wireless Personal Area Networks : WPANs). Sa très faible consommation électrique et ses coûts de production très bas en font une candidate idéale pour la domotique ou les réseaux de capteurs. *Zigbee* représente le prolongement de la norme HomeRF (Home Radio Frequency) qui a, depuis son lancement en 1998, été dépassée par le Wi-Fi. Les débits autorisés sont relativement faibles, entre 20 et 250 Kbits/s, mais c'est véritablement sa très faible consommation énergétique qui en fait son atout principal. Nous présentons sur la Figure 2.2 le positionnement de *Zigbee* par rapport aux autres types de technologies sans fil.

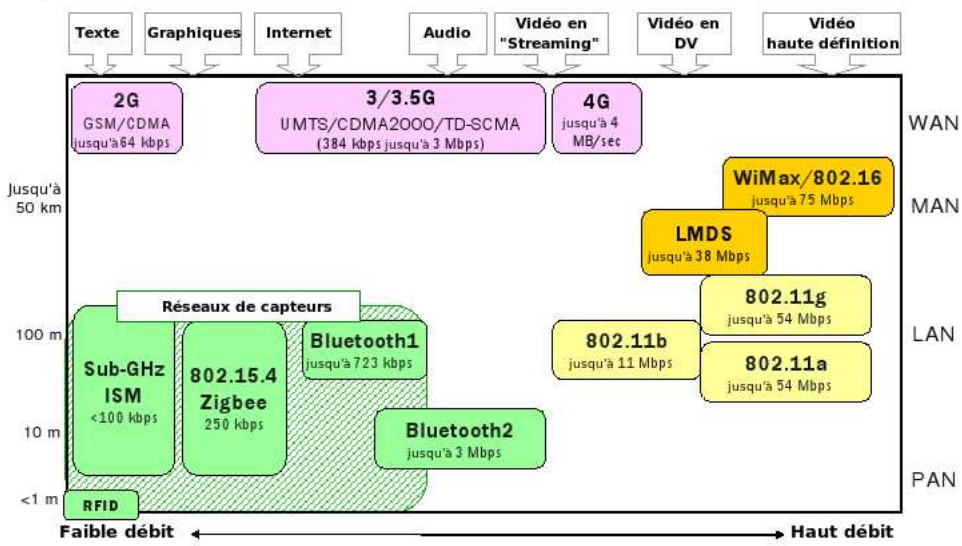


FIG. 2.2*: Les capacités des technologies sans fil*

Zigbee fonctionne sur la bande de fréquences des 2,4 GHz et sur 16 canaux. Sa portée était au début d'une dizaine de mètres, elle est désormais de 100 à 300 mètres. La spécification *ZigBee* propose une pile protocolaire propriétaire et légère. Elle s'appuie sur la norme IEEE 802.15.4 pour les couches physique et liaison et propose ses propres couches supérieures (réseau, etc.). La différence entre *ZigBee* et la plupart des autres WPAN se situe au niveau de l'utilisation du médium. *ZigBee* est optimisé pour une faible utilisation du médium sans fil partagé par tous, par exemple 0,1% du temps. Typiquement, un module émetteur/récepteur *ZigBee* occupera le média pendant quelques millisecondes en émission, attendra éventuellement une réponse ou un acquittement, puis se mettra en veille pendant une longue période avant l'émission suivante (on parle de somnolence). Cette nécessité introduit des problématiques de recherche intéressantes, notamment au niveau des couches liaison avec temporisation et stockage des messages, accès original au média et dans le cadre de la problématique réseau, le routage avec respect de contraintes énergétiques. La norme IEEE 802.15.4 prévoit deux topologies : étoile (star) où tous les noeuds communiquent avec un noeud central appelé coordinateur et point à point (peer to peer) où tous les noeuds à portée radio peuvent communiquer ensemble sans hiérarchie. La Figure 2.3 illustre la description de la pile protocolaire *Zigbee*.

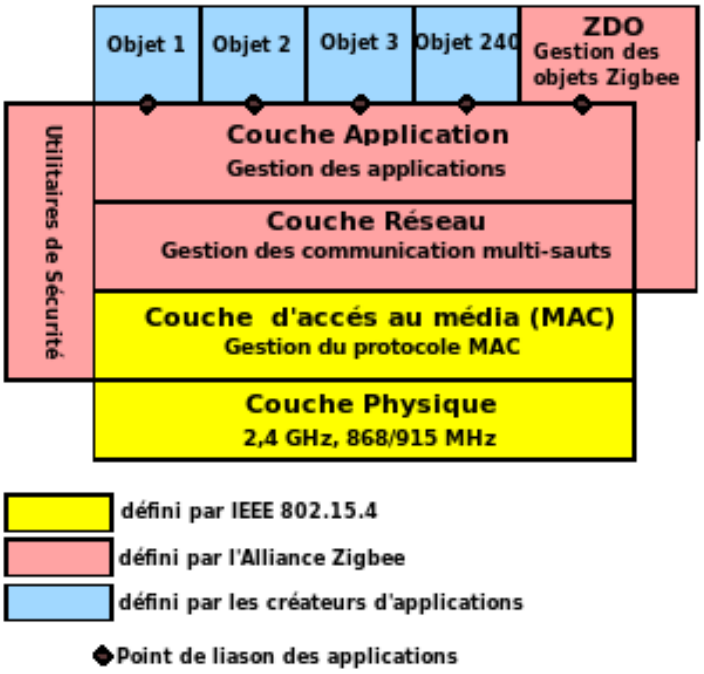


FIG. 2.3*: Pile protocolaire* Zigbee

– La couche physique ou *"Physical Layer"* est chargée de la transmission effective des signaux électriques ou optiques entre les interlocuteurs et définit les caractéristiques radio. Son service est typiquement limité à l'émission et la réception d'un bit ou d'un train de bit continu (notamment pour les supports synchrones comme la fibre optique). Cette couche est chargée de la conversion entre bits et signaux électriques ou optiques. Cette couche spécifie le mode de liaison entre les noeuds. Dans le cas des réseaux sans fil, cela va consister à la reception et à l'envoi d'ondes porteuses. Les conditions de transmission sont variables dans le temps, car le rapport signal/bruit fluctue en fonction du temps. Ceci est dû aux problèmes présentés par le canal sans fil : les effets des accès multiples, d'évanouissement, du bruit, des interférences et de la mobilité. La couche physique supporte diverses fonctionnalités comme la sélection du canal ou l'estimation de la qualité des liens.
– Le protocole MAC va définir en résumé les modalités d'accès à la communication inter-nodales. La couche d'accès au média ou *"MAC Layer"* spécifie comment les données sont expédiées entre deux noeuds/routeurs distant d'un seul saut. Elle est responsable du multi-

plexage des données, du contrôle d'erreurs, de l'accès au media. Cette couche est fortement responsable du protocole d'accès au média, le protocole MAC et du contrôle des erreurs. Le protocole MAC va définir la création de l'infrastructure du réseau et le partage des ressources entres les entités du réseau. Il va autoriser ou pas les phases de communication par différents mécanismes de synchronisation, par le biais d'un système de balise définisssant les phases d'activité des entités avec une focalisation justifiée sur l'augmentation de la durée de vie du réseau. La littérature propose de nombreux travaux dans ce domaine référencés de manière précise par [Demirkol et al., 2005].

– La couche réseau ou *"Network Layer*" permet de gérer l'adressage et le routage des données, c'est-à-dire leur acheminement via le réseau. Déterminer un chemin est une tâche complexe réalisée par des protocoles dédiés dont le rôle est de découvrir la topologie du réseau et d'en déduire la meilleure route. Les protocoles de routage se différencient par les critères de choix des routes.
– La couche Application ou *Application Layer* assure l'interface avec les applications telles que les diodes électroluminescentes ou les dispositifs de capteurs. C'est par l'intermédiaire de cette couche de la pile protocolaire Zigbee que les développeurs peuvent créer des applications.

Nous venons de présenter les différents composants d'un capteur sans fil avec la définition de la pile protocolaire *Zigbee* basée sur la norme IEEE 802.15.4. Nous avons vu que la communication sans fil, dans le cadre d'une architecture sans fil de faible portée, nécessite l'utilisation de relais pour acheminer les données de la source vers la destination. Cette particularité dans la communication est gérée par la couche réseau par le biais du protocole de routage. Le protocole de routage est important, dans les réseaux de capteurs sans fil, car c'est ce dernier qui détermine le meilleur chemin pour le transfert de données. Mais ce protocole supporte d'importantes contraintes, pour réaliser de manière efficace son objectif de routage, que nous identifions dans la partie suivante.

2.2 Notion de routage dans les réseaux de capteurs sans fil

La communication dans les réseaux de capteurs est soumise à divers phénomènes qui caractérisent les liaisons utilisant les ondes radio. Le plus connu est la forte atténuation du signal avec la distance, qui empêche deux noeuds trop éloignés l'un de l'autre de communiquer ensemble et force l'utilisation de mécanismes multi-sauts. Pour fonctionner correctement, un réseau requiert deux fonctions :

1. le routage, dont le but est de trouver un chemin, et le cas échant de trouver le meilleur chemin possible,
2. le transport, qui consiste à acheminer les messages le long d'un chemin prédéfini.

La diffusion (*broadcast*) est une manière d'acheminer les messages, de type *un vers tous*, dans laquelle un noeud source désire transmettre un message à l'ensemble du réseau. Aucun routage n'est nécessaire pour effectuer une diffusion puisqu'aucune route n'est requise. Les applications de cette opération sont nombreuses, telles la découverte de routes, de l'architecture du réseau, la découverte de services, le lancement d'alertes au sein du réseau, la synchronisation ou encore la dissémination d'informations pour un réseau de capteurs. La diffusion est donc un processus dont l'efficacité est primordiale pour le bon fonctionnement du réseau. La manière la plus simple pour effectuer une diffusion est connue sous le nom de diffusion aveugle (*flooding* ou inondation). Son principe est le suivant : chaque entité recevant pour la première fois le message à diffuser, réemet celui-ci à destination de ses voisins. Si le réseau est connexe (il existe un chemin entre la source et n'importe quel autre hôte) et que l'on suppose l'absence de collisions, alors ce processus aboutit à une couverture complète du réseau. Malheureusement cet algorithme très simple n'est pas efficace car il requiert la participation de tous les hôtes, alors que cela n'est pas toujours nécessaire. Il conduit à une grande quantité de messages redondants et d'énergie gaspillée.

Pour optimiser la participation des entités du réseau, il faut avant tout préciser que la communication peut être classée selon deux catégories devant être supportées par les différents protocoles de routage :

– application : la communication d'application est relative au transfert d'informations collec-

tées par les senseurs avec pour but d'informer l'utilisateur sur le phénomène. Cette catégorie peut être également divisée en deux avec la notion de modèle coopératif et non-coopératif. Dans un modèle coopératif, les entités communiquent entre elles dans le but de satisfaire l'objectif de l'utilisateur. Dans un modèle non-coopératif, les noeuds ne collaborent pas dans le processus de diffusion de l'information ;

– infrastructure : la communication d'infrastructure définit les messages échangés dans la mise en place du réseau ; elle est réalisée par le biais de transfert d'informations relatives à la structure du réseau. Le but de ces communications est de définir l'épine dorsale du réseau pour le processus d'acheminement des données.

2.2.1 Principes de communication

Lorsque deux entités ou noeuds d'un réseau veulent communiquer ensemble, deux situations sont possibles : soit ils sont voisins et peuvent directement échanger des messages, soit ils sont trop éloignés l'un de l'autre, auquel cas les messages doivent être retransmis de proche en proche. Une solution simple à ce problème, comme nous l'avons dit précédemment, peut être d'utiliser la diffusion pour envoyer les messages à tout le réseau, et donc en particulier à l'objet auquel ils sont destinés. Toutefois, une telle solution utilise énormément les ressources du réseau et mène rapidement à sa congestion lorsque plusieurs transmissions sont en cours. Dans un souci d'efficacité, il est préférable que les messages soient routés. Le routage consiste à découvrir un chemin entre les deux noeuds donnés, afin de ne mobiliser que les noeuds intermédiaires réellement nécessaires à l'acheminement des messages. Ces derniers sont ensuite transportés grâce à des communications le long d'un chemin sélectionné. Le processus de routage au sein des réseaux de capteurs se fait essentiellement dans le sens de communication des capteurs vers la station de base, comme illustré sur la Figure 2.4.

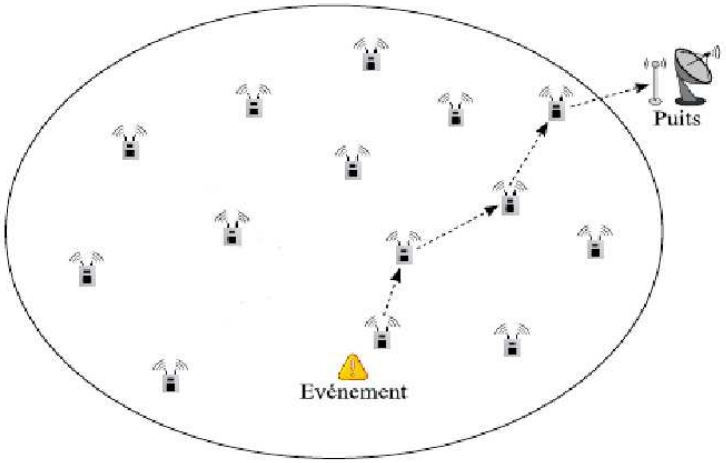


FIG. 2.4*: Architecture de réseau de capteurs sans fil*

Le routage doit donc assurer un transport efficace des informations collectées vers la station de base. Cependant le routage dans les réseaux de capteurs sans fil va être soumis à des contraintes de nature différente. Ces contraintes sont nombreuses et il est nécessaire de les identifier.

2.2.2 Identification des contraintes du routage

Nous allons voir en quoi les conditions de routage dans les réseaux de capteurs sans fil sont dépendantes de plusieurs contraintes qui les font se distinguer significativement des réseaux traditionnels sans fil.

Premièrement, dans beaucoup d'applications des réseaux de capteurs sans fil, les noeuds sont généralement statiques après le déploiement, excepté pour certains noeuds mobiles. Les noeuds dans les réseaux traditionnels sans fil sont libres de bouger, ce qui entraîne un changement imprévisible et fréquent de topologie. C'est l'une des différences majeures entre les réseaux de capteurs sans fil et les réseaux traditionnels sans fil.

Deuxièmement, du fait du nombre important de capteurs utilisés dans une application, il est impossible de construire un schéma d'adressage global pour le déploiement d'un réseau de capteurs sans fil, aux vues du nombre d'entités et du taux de maintenance élevé des adresses. Aussi,

le traditionnel protocole d'adressage IP ne peut pas être appliqué aux réseaux de capteurs sans fil comme le précise [Akkaya et Younis, 2006].

Troisièmement, en contraste avec les communications typiques des réseaux (client-serveur), toutes les applications nécessitent un flux de données collectées, des sources vers une station de base ou réservoir. L'orientation de ce flux de données est importante dans la conception des protocoles de routage car ces derniers auront pour but de favoriser cet aiguillage lors de l'envoi des données.

Quatrièmement, les noeuds sont intrinséquement liés aux contraintes énergétiques, de calcul et de capacité de stockage. Les protocoles de routage doivent avoir une attention particulière sur la gestion des ressources.

Cinquièmement, il est raisonnable de considérer que les noeuds sont spécifiques à une application comme le précise [Estrin et al., 1999a]. Le nombre important des applications [Xu, 2002] dans le domaine des réseaux de capteurs sans fil conduit également à plusieurs restrictions au niveau de l'énergie, de la puissance de calcul et de la limitation de la bande passante des liens sans fil connectant les noeuds. Par conséquent, le routage est dépendant de l'application et doit tenir compte des particularités de cette dernière.

Nous voyons donc que le routage est influencé par ces multiples défis. Soumis à différentes contraintes, un protocole de routage doit répondre à certains objectifs pour permettre une communication efficace. Dans le cadre de notre problématique, le réseau de capteurs sans fil, qui permettra de prévoir, détecter et suivre le phénomène feu de forêt, doit utiliser un protocole de routage répondant à certains critères que sont :

- la tolérance aux fautes : les noeuds peuvent faillir par le fait du manque d'énergie, des dommages physiques (panne matérielle) ou d'interférence environnementale (destruction par le feu). Les défauts de certains noeuds ne doivent pas interferer sur la tâche globale de l'ensemble du réseau. Le protocole doit avoir la possiblité de s'adapter à ces défaillances ;
- l'homogénéité des noeuds : dans le cadre de notre problématique, nous désirons que tous les noeuds possédent le même rôle dans le réseau. L'homogénéité des noeuds doit autoriser n'importe quelle entité du réseau à pouvoir réaliser des tâches de collecte de données et de

routage. Le protocole doit prendre en compte cette considération ;

– la consommation énergétique : les noeuds utilisent leur faible ressource énergétique pour le calcul et la transmission des données dans leur environnement sans fil. Par conséquent les formes d'économie d'énergie dans la communication sont essentielles. La durée de vie du capteur montre une forte dépendance vis-à-vis de la batterie. Dans un routage multi-sauts, chaque noeud joue un double rôle : transmetteur de données et routeur. Le protocole doit s'attacher à remplir ces tâches à économiser la ressource énergétique pour suivre le feu sur long terme ;
– le mode de report des données et qualité de service : le mode de report de données traduit le processus d'un noeud pour envoyer les informations collectées à la station de base ; les données collectées par ces réseaux sont dépendantes du type d'application et surtout du temps de report de ces données ou latence (temps entre la collecte d'informations et l'arrivée des informations sur la station de base). Le report des données peut être conduit par le temps (continu), par les événements ou par la demande. Le mode "dirigé par le temps" est approprié pour les applications qui requièrent des demandes de données périodiques. En tant que tels, les capteurs passeront périodiquement de leur mode de détection au mode transmission. Dans le mode "dirigé par les événements ou par la demande", les capteurs réagissent immédiatement à des changements soudains dans la valeur des données collectées en raison de l'occurence d'un certain événement ou d'une demande générée par la station de base. Tel quel, il est approprié pour les applications où le temps jouent un rôle crucial, comme par exemple la détection précoce de conditions favorisant l'éclosion d'un feu de forêt. Le protocole de routage est grandement influencé par le mode de report des données, qui va jouer directement sur la consommation énergétique et la stabilité du routage ;
– extensibilité : le nombre de noeuds déployés dans une zone de détection peut aller de la centaine au millier. Les algorithmes de routage doivent être capable de travailler sans considérations vis-à-vis du nombres d'entités ;
– le déploiement des noeuds : le déploiement est dépendant de l'application et affecte les performances du protocole de routage. Il peut être déterministe ou aléatoire. Dans une configu-

ration déterministe, les capteurs sont placés manuellement et dans un déploiement aléatoire, les noeuds sont déposés de manière indéterminée. Le protocole de routage doit pouvoir s'adapter à ces deux types de déploiements ;

– la dynamique du réseau : certaines architectures de réseau considérent que les noeuds sont stationnaires. Cependant, la mobilité de la station de base et de certains noeuds existent dans certaines applications et cela le protocole doit en tenir compte.

Nous avons identifié les caractéristiques que nous recherchons dans un protocole de routage. Les algorithmes de routage sont nombreux et il existe un nombre conséquent de travaux. Différentes taxonomies proposent une classification des protocoles par famille. Nous proposons d'étudier ces différentes taxonomies et de distinguer la famille de protocole permettant le routage le plus efficace (respect des caractéristiques recherchées) dans le cadre de notre étude sur les feux de forêt.

2.2.3 Taxonomie des protocoles de routage

De nombreux algorithmes de routage sont proposés pour les réseaux de capteurs sans fil prenant en considération les paramètres cités précédemment. Selon [Al-Karaki et Kamal, 2004], les protocoles de routage peuvent être classés selon trois catégories : routage dit "plat", routage hiérarchique, routage basé sur la localisation .

Dans le routage dit "plat", tous les noeuds ont un rôle ou une fonctionnalité équivalent. Tous les noeuds vont ensuite collaborer pour réaliser une tâche. Comme nous l'avons dit précédemment, il est difficile d'attribuer un identifiant global du fait du nombre important de noeuds. Cette considération a conduit au routage centré sur les données, où la station de base envoie des questions dans certaines régions et attend les données des noeuds localisés dans ces régions. Cette famille de routage favorise, de manière assez fréquente, le report de données "dirigé par les événements".

Dans un routage hiérarchique ou *clustering*, cela est différent. Les noeuds ne jouent pas tous le même rôle, basé sur des notions de *leader* lors des transmissions. En effet, la première motivation étant l'économie d'énergie, certains noeuds vont avoir un rôle continu ou alternatif dans l'agrégation et la transmission des données. Ainsi le réseau, utilisant un routage hiérarchique, sera

constitué d'entités n'ayant pas toutes le même rôle, avec une relation maître-esclave.

Dans le routage basé sur la localisation, l'adressage se fait selon la localisation géographique du noeud. Des coordonnées peuvent être obtenues par l'échange des informations géographiques entre les voisins. La localisation des noeuds se fait de manière assez fréquente par le biais d'un module GPS.

Plusieurs travaux ont eu pour but de présenter un classification des protocoles de routage existants [Ácc et Buttyán, 2006][Akkaya et Younis, 2006][Al-Karaki et Kamal, 2004]. L'étude la plus complète est apportée par [Al-Karaki et Kamal, 2004] et nous propose la classification suivante :

Routage dit "plat"	**Routage hiérarchique**	**Routage basé sur la localisation**
Direct Diffusion[Intanagonwiwat et al., 2000]	**LEACH**[Heinzelman et al., 2000]	**GAF**[Xu et al., 2001]
SPIN[Heinzelman et al., 1999]	**PEGASIS**[Lindsey et Raghavendra, 2002]	**GEAR**[Yu et al., 2001]
COUGAR[Yao et Gehrke, 2002]	**TEEN**[Manjeshwar et Agarwal, 2001]	**MFR, DIR, GEDIR**[Stojmenovic et Lin, 1999]
Rumor Routing[Braginsky et Estrin, 2002]	**MECN**[Rodoplu et Meng, 1999]	**GOAFR**[Kuhn et al., 2003]
MCFA[Ye et al., 2003]	**SOP**[Subramanian et Katz, 2000]	**SPAN**[Chen et al., 2002]
Gradient Based Routing[Schurgers et Srivastava, 2001]	**Sensor Agregate Routing**[Fang et al., 2003]	
IDSQ/CADR[Chu et al., 2002]	**VGA**[Al-Karaki et al., 2004]	
ACQUIRE[Sadagopan et al., 2003]	**HPAR**[Li et al., 2001]	
EAR[Shah et Rabaey, 2002]	**TTDD**[Ye et al., 2002a]	
Random Walks[Servetto et Barrenechea, 2002]	**APTEEN**[Manjeshwar et Agerwal, 2002]	
Reliable Route Protocol or Xmesh[Woo et al., 2003a]		

TAB. 2.1*: Taxonomie des protocoles de routage selon [Al-Karaki et Kamal, 2004]*

Nous ne proposons pas dans cette partie de présenter les différents protocoles de routage. Nous proposons d'étudier les différents familles de protocoles (dit "plat", hiérarchique et basé sur la localisation) et de réaliser un tri qui déterminera la famille de protocole réellement applicable dans notre contexte d'étude.

2.2.4 Le protocole de routage dans la problématique feux de forêt

Le phénomène destructeur que représente les feux de forêt, peut avoir un impact plus ou moins important sur les réseaux de capteurs sans fil. La surveillance d'un tel phénomène engendre une

destruction des entités du réseau, rendant la structure et les communications du réseau défaillantes. Une réflexion est nécessaire pour déterminer quelle est la famille de protocole la mieux adaptée à la problématique des feux de forêt. Nous avons vu que les protocoles de routage peuvent être classée en trois catégories selon [Al-Karaki et Kamal, 2004] :

- dans le routage dit "plat", tous les noeuds ont un rôle ou une fonctionnalité équivalent.
- dans le routage hiérarchique ou *clustering*, les noeuds ne jouent pas tous le même rôle ; certains noeuds *leader* vont gérer l'agrégation et la transmission des données de noeuds esclaves ;
- dans le routage basé sur la localisation, la principale caractéristique est que les noeuds utilisent la localisation géographique pour l'aiguillage du flux d'informations ; pour fournir la position des noeuds, le module GPS est souvent utilisé.

Dans le cas des feux de forêt, le routage des informations entre les noeuds implique une adaptation au niveau de la structure soumise à une destruction massive. En effet, comme nous l'avons dit précédemment, nous recherchons une famille de protocole tolérante à une défaillance d'une partie du réseau. Cette défaillance peut être due au phénomène étudié (dans notre cas ce sera un capteur brûlé) ou due à une défaillance technique.

Dans le cas d'un routage hiérarchique, le réseau présente des points sensibles. En effet, les noeuds centraux ou *leader* sont les points faibles de l'architecture. Ces noeuds, qui ont un rôle central dans le réseau, collectent les données de capteurs esclaves qui sont à portée radio. Le principal avantage est l'économie énergétique. Cependant, dans le cas particulier d'un feu de forêt, plusieurs *leader* peuvent être détruits. Ces derniers ne peuvent plus transmettre les informations et par conséquent orientent le réseau vers une défaillance générale. Cette famille de protocole ne répond pas à nos exigences d'homogénéité des noeuds dans le réseau et de tolérance aux fautes.

Dans le cas du routage géographique, la technique de routage est basée sur la position des noeuds. Selon[Ermel, 2004], voici les prérequis pour effectuer un routage géographique :

- les noeuds possèdent tous un moyen de localisation : soit un système natif comme le GPS soit un système logiciel ;
- un noeud source connaît toujours la position du noeud destinataire. Pour ce faire, soit tous

les noeuds connaissent les positions initiales de tous les noeuds, soit un service de localisation doit être utilisé.

Le routage géographique nous conduit à soumettre deux hypothèses quant à son application dans la problématique feux de forêt :

1. l'utilisation du GPS entraîne un surplus de consommation donc cela fait diminuer la durée de vie des entités [Crossbow-Technology, 2006] ;
2. l'utilisation d'un GPS augmente le taux de défaillance dans le réseau en admettant que le module de localisation puisse lui aussi être défaillant.

Cette famille de protocole de routage ne répond pas à nos attentes en terme d'économie d'énergie et de tolérance aux fautes.

Le routage "dit plat" regroupe les protocoles autorisant une homogénéité des noeuds dans le réseau. Les noeuds utilisant cette technique de routage, seront à la fois collecteur d'informations et routeur. Concernant l'aiguillage de ces informations , le routage dit "plat" ne se base pas sur des coordonnées géographiques fournies par un GPS ; cette famille utilise d'autres paramètres de routage, comme le nombre de sauts jusqu'à la station de base ou la capacité énergétique des noeuds du réseau.

Cette famille de protocole est intéressante car elle regroupe certaines caractéristiques recherchées. En particulier, nous relevons l'homogénéité des noeuds, la prise en compte du niveau énergétique, la tolérance à certaines défaillances (pas d'utilisation de module GPS) et le report de données "dirigé par les événements".

Pour illustrer la notion de défaillance, que nous considérons comme essentielle, nous proposons la Figure 2.5. En rouge foncé, nous signalons les noeuds détruits ("brûlés") durant un feu et en jaune clair, les noeuds dont le système de localisation, le module GPS par exemple, est défaillant.

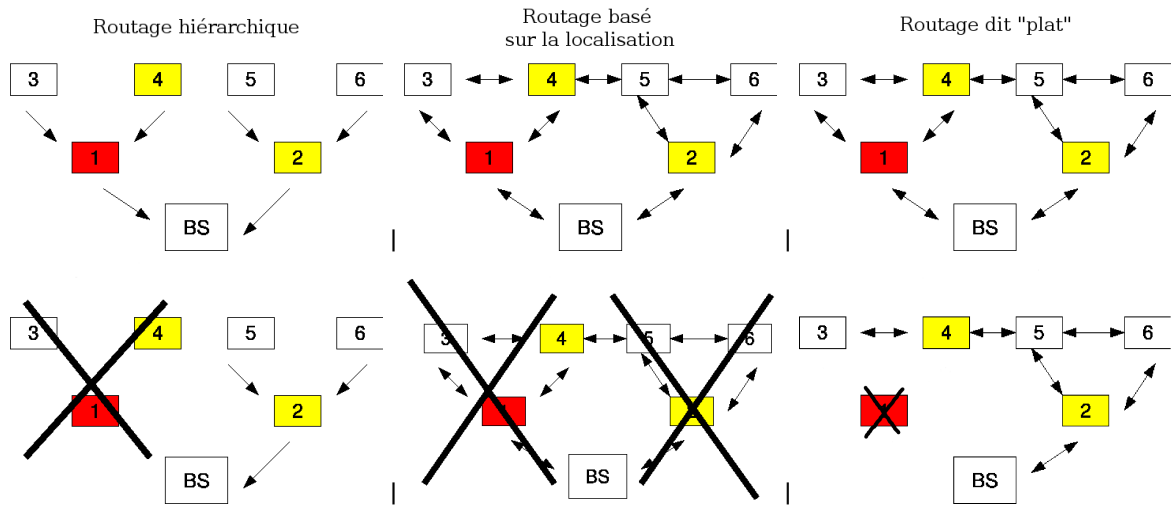


FIG. 2.5*: Les techniques de routage soumises à une double défaillance*

Nous voyons par ce simple exemple que dans le cas du routage hiérarchique, une partie des noeuds esclaves gérés par le noeud *leader* détruit par le feu, ne pourront plus transmettre les informations à la station de base. En effet, dans le cas du routage hiérarchique, les noeuds esclaves communiquent exclusivement avec le noeud *leader* et la destruction de ce dernier est fatale pour une partie du réseau. Dans le cas du routage géographique, les noeuds ne sont pas soumis à la domination d'un noeud, cependant la défaillance de trois noeuds, un détruit par le feu et deux autres avec un module GPS défaillant , font évoluer la structure vers un stade critique ne permettant plus aucune transmission d'informations. Seule la technique dite "plat", où tous les noeuds sont égaux et ne basant pas leur routage sur des données géographiques, peut permettre la survie de la structure. En effet, nous voyons sur la Figure 2.5 que le réseau ne peut pas être pris à défaut sur une double défaillance, feu et panne du GPS. Les noeuds, ne centralisant pas les données et n'utilisant pas le module GPS pour router les informations, peuvent continuer à transmettre leurs informations en utilisant justement le noeud possédant un module GPS défaillant.

C'est pour cette raison de robustesse que nous préférerons donc, dans le cadre de notre problématique des feux de forêt, une étude sur les techniques issues de la famille des protocoles dit " plat".

Dans la réalité, il n'est naturellement pas concevable de déployer plusieurs capteurs dans un feu et d'en étudier le comportement. La solution est de simuler un réseau dans les conditions d'un incendie et d'analyser le comportement. Pour cela, il existe plusieurs approches que nous nous proposons de détailler et de comparer pour trouver un outil étant capable de répondre à notre problématique.

2.3 Les outils d'analyse des réseaux de capteurs sans fil.

Il existe des outils, tels que NS-2[NS2, 1995], OMNeT++[Omnet++, 1992], pour étudier les réseaux sans fil. Cependant ces derniers ne sont pas axés sur une recherche propre aux comportements de réseaux de capteurs sans fil qui doivent être analysés différemment, au vu des contraintes existantes et des actions à réaliser. Nous allons rechercher, dans les approches existantes, des fonctionnalités qui vont nous permettre d'étudier un réseau de capteur dans le cadre de notre problématique. Ces fonctionnalités sont les suivantes :

1. **Environnement** : le modèle doit intégrer la possibilité de générer des scénarii environnementaux ; dans le cadre de notre étude, nous désirons pouvoir étudier un scénario de feux de forêt.
2. **Modèle énergétique** : l'énergie, aspect crucial dans la durée de vie des capteurs, doit pouvoir être représentée de manière précise.
3. **Composants matériels** : le modèle doit pouvoir prendre en compte tous les composants d'un capteur et avoir la capacité d'étudier le comportement de chacun.
4. **Topologie ou déploiement** : le modèle doit favoriser l'étude du déploiement du réseau et les obstacles du monde réél doivent pouvoir être pris en compte.

Dans cette partie, nous proposons un survol des approches existantes en y recherchant les fonctionnalités citées précédemment.

2.3.1 Le modèle selon TOSSIM (TinyOS SIMulator)

Le modèle de capteur sans fil proposé par l'approche TOSSIM [Levis et al., 2003] définit les composants essentiels de la plate-forme MICA2 de la société *Crossbow Technology*. De ce modèle découle un simulateur à évènements discrets basé sur le système d'exploitation embarqué TinyOS (*Tiny Operating System*). Ce simulateur émule les composants matériels du capteur. TinyOS est un système d'exploitation pour les sytèmes embarqués, plus particulièrement orienté sur les réseaux de capteurs sans fil. Il posséde un modèle de programmation basé sur les composants, implémenté en nesC, un langage dérivé du C. TinyOs ne peut se définir comme un système d'exploitation (*SE*) au sens strict. Il représente en fait un cadre de programmation pour les systèmes embarqués et une série de composants, capables de construire une application spécifique au *SE*. Le but avoué de ce simulateur est de fournir un support de test pour les applications TinyOS dépendantes. Il permet de développer, tester et analyser des algorithmes qui seront incorporés au système embarqué. Il fournit une simulation très fidèle des applications TinyOS, en observant le fonctionnement comportemental du système d'exploitation à un niveau très bas. Il capture le comportement et les interactions de centaines de noeuds TinyOS en utilisant le bit comme niveau de description des données. Une extension du simulateur appelée PowerTOSSIM[Shnayder et al., 2004] a permis d'intégrer un modèle de consommation énergétique en isolant la consommation de chaque composant. Cela se concrétise par l'ajout d'un module, surveilllant l'utilisation de la ressource énergétique.

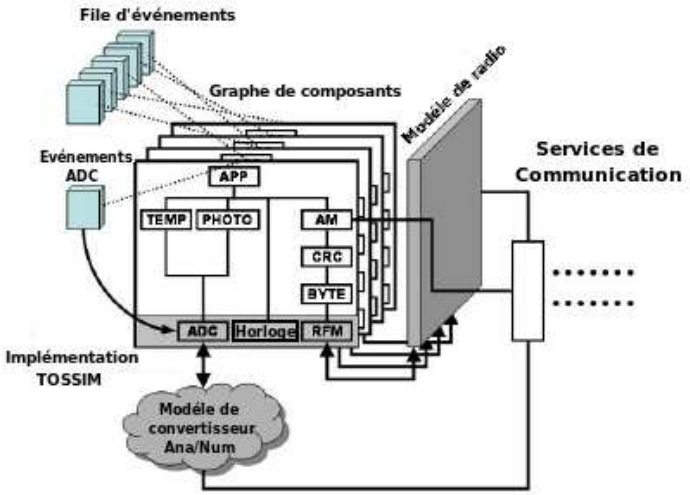


FIG. 2.6*: Le simulateur TOSSIM [Levis et al., 2003]*

L'architecture de TOSSIM est constituée de cinq parties comme illustré sur la Figure 2.6 :

- un support pour la compilation de graphes de composants TinyOS ;
- une file d'événements discrets ;
- un petit nombre de composants matériels TinyOS réimplémentés ;
- un modèle de radio et de convertiseur analogique/digital qui correspondent à une émulation des composants matériels ;
- et une interface pour les services de communication pour les programmes externes interagissant durant la simulation, en particulier pour l'outil de visalisation et l'interface de programmation TinyViz.

TOSSIM profite de la structure TinyOS et réalise une compilation complète du sytème pour obtenir une simulation à événements discrets directement à partir du graphe de composants du système d'exploitation. Il exécute le même code que celui utilisé sur le matériel des capteurs sans fil. TOSSIM va traduire les signaux du matériel, par le biais de ses composants à l'intérieur de la file d'événements. La file d'événements du simulateur délivre les signaux qui dirige les exécutions de l'application TinyOS.

TOSSIM est un outil de simulation efficace. Non seulement, il permet de simuler un grand nombre de capteurs de manière précise, mais de plus il permet le développement et le test d'applications basées TinyOS. Cependant, dans notres cas, il est difficile d'utiliser cet outil pour introduire des conditions environnementales. Il est possible de tester le fonctionnement des capteurs, mais TOSSIM n'offre pas la possibilité de travailler sur un scénario environnemental de type feu de forêt. La possibilité d'étudier différents types de déploiement de réseau n'est pas un aspect pris en compte par cette approche. Si il est possible de travailler sur les applications, il est difficile de définir de nouveaux composants matériels de manière simplifiée car ils sont complètement dépendants de TinyOS.

2.3.2 Le modèle selon SensorSim

L'approche des réseaux de capteurs selon SensorSim [Park et al., 2000] fournit un cadre de modélisation et de simulation des réseaux de capteurs. Le coeur de SensorSim est basé sur le simulateur très populaire NS-2[NS2, 1995], lui fournissant une extension dans le but de représenter le comportement des réseaux de capteurs.

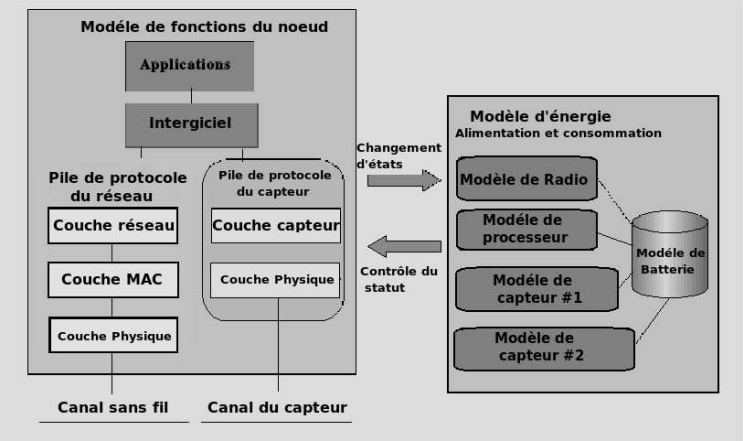


FIG. 2.7*: Le modèle de capteur sans fil dans SensorSim [Park et al., 2000]*

La Figure 2.7 représente le modèle utilisé pour la simulation du réseau de capteur. En réalité il y a deux types de modèles :

1. le modèle de fonctions du noeud ; il décrit les modules logiciels permettant le fonctionnement des dispositifs constituant le noeud ; les fonctions décrites regroupent la gestion du réseau, la partie logiciel, les applications utilisateurs et les activités des capteurs ;
2. le modèle d'énergie fait apparaître différents éléments capables de consommer de l'énergie et implicitement d'en produire : le processeur, la radio et le sensorboard.

Les deux modèles peuvent être vu comme des couches parallèles simulant la partie matérielle et la partie logicielle du noeud. Dans le modèle de capteurs selon l'approche SensorSim, le modèle logiciel utilise le modèle d'énergie pour effectuer ses actions. Les auteurs nous présentent également une notion appelée *canal du capteur*. Par analogie avec le *canal sans fil*, représentant le mode de communication aérien des ondes radio, le *canal du capteur* peut être vu comme l'environnement où les senseurs peuvent détecter les événements. Les auteurs soulignent plus particulièrement un modèle de consommation énergétique au niveau de la radio, avec la prise en compte de diverses phases, telles que la phase de transmission, de réception ou de mise au repos du capteur. La possiblité de gestion des divers protocoles que ce soit de routage ou bien de l'accés au média est vraiment intéressante. La décomposition des composants (processeur, radio, modèle énergétique, etc.), proposée par le modèle SensorSim, est pertinente. La décomposition des phases de communication radio offre également plus de lisibilité dans le suivi des processus de transmission et de réception.

Cependant, SensorSim, basé sur le simulateur NS-2, est limité dans le cadre d'évolution des composants par rapport à la complexité d'implémentation du simulateur NS-2. Il apparait également difficile d'implémenter un modèle environnementale et d'étudier différents déploiements de réseau. En effet, nous désirons étudier un scénario de feu de forêt en fonction de différentes topologies.

2.3.3 Le modèle selon SENS

Le modèle de capteur proposé SENS [Sundresh et al., 2004] revendique une indépendance face à la plateforme. Il possède une architecture modulaire, avec des composants modifiables qui représentent les applications, les communications dans le réseau et l'environnement physique. La

structure du simulateur consiste en une interaction de plusieurs noeuds simulés avec un composant dédié à l'environnement. Chaque noeud est constitué de trois composants nommés *Application*, *Réseau* and *Physique* comme illustré sur la Figure 2.8 :

- le composant *Application* simule l'exécution de la partie logicielle du noeud. Il communique avec le composant *Réseau* pour envoyer ou recevoir des paquets de données et communique avec la couche *Physique* pour lire les valeurs renvoyées par les capteurs ;
- le composant *Réseau* simule les fonctions d'envoi et de reception des données du capteur sans fil ;
- le composant *Physi*que représente les modèles de senseurs physiques, les déclencheurs d'action et les interactions avec l'environnement. Il permet aussi de modéliser une utilsation de la ressource énergétique ;
- le composant *Environnement* a pour but de fournir un modèle environnemental permettant de simuler une interaction avec les éléments de la couche physique.

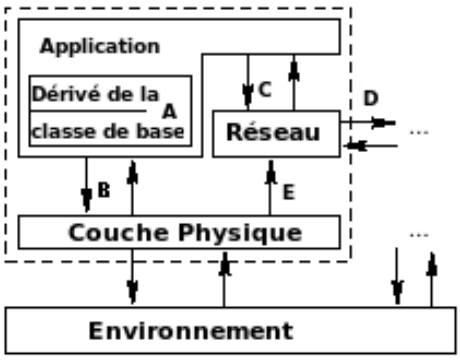


FIG. 2.8*: Les composants du simulateur SENS[Sundresh et al., 2004]*

Une des particularités intéressantes dans la simulation avec SENS est la possibilité de prendre en compte divers scenarii sur l'environnement réél du réseau, c'est à dire les contraintes physiques qui peuvent empêcher les communications dans le réseau (mur, végétaux), que l'on peut regrouper sous le terme générique d'obstacles. Cette disposition autorise une réflexion sur le déploiement des réseaux de capteurs.

L'approche proposée par SENS, dans ses aspects d'indépendance face à la plate-forme, la possibilité de travailler sur la topologie, la modularité de l'architecture et la rapidité d'implémentation, est réellement intéressante.

Cependant la définition des composants n'est pas assez fine et rend l'approche incomplète. En effet le modèle ne prend pas en compte la consommation énergétique, point crucial dans le contexte des réseaux de capteurs. De plus, la définition des mécanismes de communication n'est pas clairement établie notamment dans la prise en compte dans le processus d'accès au média.

2.3.4 Le modèle selon SWAN (Simulator for Wireless Ad-hoc Network)

Le projet SWAN [Liu et al., 2001] représente la jonction de deux parties : d'un côté un simulateur extensible, DaSSF développé par l'Université de Dartmouth et de l'autre un logiciel de routage Wirokit pour les réseaux sans fil *ad-hoc*, issu de l'experience de BBN. Le simulateur SWAN (*Simulator of Wireless Ad-hoc Network*) offre la possibilité de simuler un réseau sans infrastructure centrale en permettant de tester en particulier les techniques de routage.

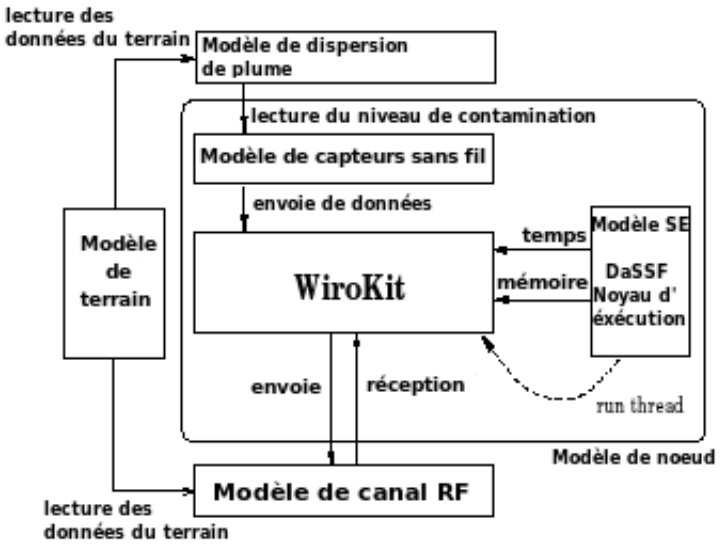


FIG. 2.9*: Le simulateur SWAN*

Le simulateur existant, basé sur le SSF (Scalable Simulation Framework), est un standard de

domaine public développé par le consortium S3[1]. Il autorise la simulation de systèmes complexes à événements discrets : c'est une simple interface pour la construction de modèles de simulation. Il est orienté processus, isolant le modélisateur des complexités de gestion de temps et de la gestion des listes d'événements. SSF est orienté objet et son interface de programmation définit cinq classes : *entity* (*il contient des variables d'états*), *process* (*définit comment les états changent*), *outchannel*, *inchannel*, *event*. L'échange des données à travers les entités se fait grâce à un canal (*channel*), qui s'organise autour d'un flux unidirectionnel d'événements entre deux entités. Le modèle, défini par SSF, est capable d'exprimer le processus que chaque sous modèle utilise pour communiquer avec les autres en se basant sur la définition des données échangées.

DaSSF (DartmouthSSF), développé par l'Université de Darmouth, est l'implémentation d'un système d'exécution de simulation conforme à l'interface de programmation de SSF. Pour résumer, DaSSF fournit principalement une colle structurale permettant une interconnexion de sous-modèles mais il fournit également l'infrastructure pour l'échange de données, et un outil pour la synchronisation de tous les composants du réseau. Pour le routage, SWAN utilise WiroKit développé par BBN Technologies, outil pour la gestion des messages de routage au niveau du réseau. WiroKit a été créé pour être portatif non seulement à travers différentes plate-formes sans fil, mais également facilement transportable sur d'autres bancs d'essai de simulation, permettant une exécution directe des algorithmes de routage au niveau de code source.

Dans la représentation du capteur, on trouve les composants physiques, l'algorithme de routage mais également le système d'exploitation exposé sur la Figure 2.9 (ce dernier n'est pas réellement nécessaire car par défaut, SWAN fournit un OS par le biais de DaSSF). Le modèle de noeud défini dans SWAN contient les composants suivants : *GPS*, *Sensor* ou *Monitor*, *router*, *IP*, *MAC*, *PHY*, décrivant chacun une action particulière dans le noeud :

- le composant *GPS* informe de la localisation du noeud et du temps ;
- *Sensor* mesure l'activité chimique et il peut également composer et transmettre des messages ;
- *Monitor* représente le réservoir d'accueil des informations et il analyse l'évolution de la

[1] S3 : Scalable Self-Organising Simulations consortium

simulation ;

- *IP* simule la couche internet protocol dans un ordinateur hôte dans le cas d'un raccordement du réseau sans fil à un réseau filaire ; il fournit des entêtes de données IP pour chaque message de noeud :
- *router* est un moteur de routage pour les réseaux ad-hoc associé avec son moteur d'envoi : ici Wirokit joue ce rôle ;
- *MAC* simule la couche d'accès au média et *PHY* représente la couche physique, mais pour l'instant ces deux objets sont vides.

SWAN fournit un modèle décrivant de manière précise les composants d'un capteur sans fil. De plus, l'avantage de SWAN est de proposer un outil indépendant dans la gestion du protocole de routage.

Cependant, le modèle ne prend pas en compte le paramètre énergétique. De plus, il semble difficile d'implémenter le modèle environnemental et de proposer l'étude de différentes topologies. Enfin, la modularité de la structure n'est pas réellement prise en compte.

2.3.5 Le modèle selon Glonemo (**Glo**bal **ne**twork **mo**del)

Le modèle de capteur proposé par l'approche Glonemo [Samper et al., 2006] utilise un formalisme décrivant les communications parallèles interprétées par des automates. Comme un automate a un nombre fini d'états explicites et que les transitions sont marquées par des conditions sur quelques variables, cela permet de tester la présence ou l'absence de signaux, l'affectation de variables et l'émission de signaux. Ces signaux sont utilisés pour la synchronisation des automates selon un mécanisme de diffusion synchrone.

Le modèle global est un ensemble de processus communiquants utilisant le ReactiveML ou Lucky. Le noeud est constitué de différentes parties :

1. un modèle de noeud, exprimant le comportement fonctionnel de l'entité ; le modèle possède une instance pour chaque noeud, et toutes ces instances sont des processus parallèles ;
2. un modèle AIR, qui représente le média sans fil pour l'échange de média sans fil à travers le réseau ;

3. un modèle environnement qui envoie des messages à la partie application dans le cadre de la détection environnementale
4. une série d'observateurs ayant pour but de vérifier les messages provenant du modèle environnement et des communications.

Suivant la Figure 2.10, le modèle de noeud est décrit comme la composition de processus parallèles décrivant les applications, le routage dans le réseau, du protocole d'accès au média ou couche MAC et une couche pour la définition des composants matériels. Dans un déroulement classique de détection de franchissement de seuil de détection, le processus *Application* envoie un message au processus *Routage* qui le transmet au processus *couche MAC* pour la transmission des informations dans l'air.

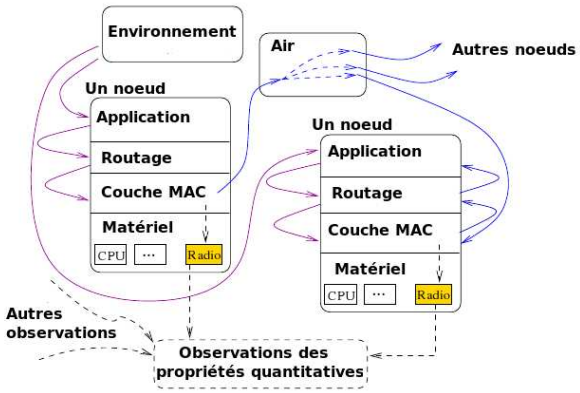


FIG. 2.10*: Représentation des noeuds dans Glonemo[Samper et al., 2006]*

L' approche proposée par Glonemo est intéressante dans la prise en compte des processus de communication entre les couches et la notion d'étude des messages échangés. La possibilté d'observer le comportement du réseau par le biais d'observateurs est appréciable.

Cependant la consommation énergétique n'est pas prise en compte globalement dans le modèle de capteur. En effet, il ne prend en compte que la consommation au niveau de la *couche MAC*. Nous rappelons que ce paramètre est essentiel dans l'étude d'un réseau de capteurs car tous

les composants consomment de l'énergie. Il est également difficile d'étudier des déploiements différents.

2.3.6 Le modèle selon ATEMU (ATmel EMUlator)

ATEMU [Polley et al., 2004] fournit une émulation des opérations de chaque noeud d'un réseau. Un émulateur est un artefact logiciel qui simulera un comportement. Il représente les actions des différents composants d'un noeud, tels que le processeur ou l'interface radio par exemple. Le but est de représenter par l'interaction des différents noeuds entre eux, une émulation du réseau de capteurs sans fil dans sa globalité. ATEMU est constitué de deux composants : l'émulateur de réseau ATEMU et le gestionnaire graphique XATDB comme représenté sur la Figure 2.11 .

A la base, ATEMU est un émulateur de logiciel pour le processeur AVR basé sur des plates-formes telles que MICA2. Dans sa conception, il inclut également d'autres périphériques caractérisant ce type de réseau comme la radio par exemple. Il peut modéliser l'exécution des noeuds et leurs interactions en utilisant une définition fine du processus de communications radio. Il offre en plus une émulation complète de la plate-forme MICA2 et fournit des résultats précis sur les opérations se réalisant à l'intérieur du système.

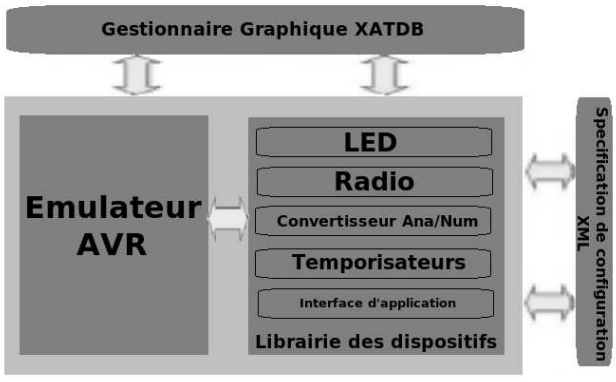


FIG. 2.11*: Les composants du simulateur ATEMU*

La partie la plus importante du simulateur ATEMU concerne la possibilité de simuler le comportement du CPU AVR comme illustré sur la Figure 2.11. Chaque instruction de l'AVR est décodée et simulée selon les spécifications du constructeur Atmel. En plus de cette particularité, ATEMU fournit divers modules de matériels pouvant être ajoutés pour définir les caractéristiques d'un capteur. Dans ce cas précis, ATEMU fournit des implémentations de presque tous les dispositifs nécessaires pour la représentation du comportement de la plate-forme MICA2 par le biais d'une librairie de composants. L'émulation de l'interface radio est un composant essentiel des capteurs sans fil. Dans son implémentation du processus de communication, ATEMU utilise une définition précise du mode de propagation des ondes radio basé sur une interface sans fil. XATDB est un *front-end* graphique, fournissant aux utilisateurs un système complet pour le débogage et la surveillance des exécutions du code. Par le biais de cet utilitaire, l'utilisateur peut visualiser et contrôler un noeud, indépendamment des autres, en fonction de chaque instruction donnée par le processeur.

L'aspect librairie de composants est un aspect vraiment intéressant dans l'approche ATEMU. De même que l'étude du comportement du processeur apporte de la précision dans l'analyse de chaque capteur.

Cependant cette approche est trop dépendante d'un type de matériel. En effet, ATEMU semble être dédié à un type de processeur. De plus l'aspect énergétique n'est pas abordé dans ces travaux. Il apparait également difficile de travailler sur l'implémentation du protocole de routage, sur la notion de déploiement des capteurs ou sur la mise en place d'un scénario environnemental, que nous recherchons systématiquement.

2.3.7 Le modèle selon Avrora (The AVR simulation and analysis framework)

En se basant sur l'approche relative au comportement du processeur, avancé par le simulateur ATEMU, Avrora [Titzer, 2004, Titzer et al., 2005] définit le fonctionnement de chaque sonde en interprétant le code machine du programme de réseau tout en maintenant l'exactitude de cycle. Il simule le fonctionnement de différents dispositifs câblés tels que le capteur, les temporisateurs, les

sondes et la radio. Avrora emploie une file d'attente d'événements pour maintenir la synchronisation entre les dispositifs et l'horloge de l'unité centrale. Ce modèle asynchrone améliore l'exécution et réduit ainsi les frais généraux de la simulation. En effet, Avrora simule l'exécution, au niveau de code machine, des applications de réseau de capteurs, écrites en n'importe quel langage, et de tous les composants de logiciel d'exploitation comme TinyOS et SOS[SOS, 2004], deux *SE* populaires de réseaux de capteurs. Avrora peut manipuler des réseaux de milliers de noeuds. Par une stratégie efficace de synchronisation des interactions entre les noeuds, il préserve les propriétés dépendantes du temps. Avrora s'assure que le comportement du réseau est non seulement représenté correctement dans le temps, mais également précis à différents rythmes à bas niveau. Une telle précision de synchronisation est une chose nécessaire pour modéliser exactement et analyser de nombreuses propriétés des réseaux de capteurs sans fil, y compris l'énergie. En effet, la consommation d'énergie dans ces réseaux est un facteur de conception crucial. Avrora fournit une base flexible de simulateur pour le modèle d'énergie à l'aide d'une application Java. Le principal avantage d'Avrora avancé par les auteurs est qu'il offre une simulation plus précise du comportement du microcontrôleur afin de détecter les failles de programme qui ne sont pas apparues lors des simulations réalisées par TOSSIM.

Avrora propose une définition plus précise du modèle ATEMU, améliorant la synchronisation entre les dispositifs et par conséquent les résultats de simulation. Il apporte une certaine plus-value en fournissant une application pour l'étude de la consommation énergétique.

Fortement dépendante du matériel, cette approche a pour but de tester le matériel et non une étude d'un réseau de capteurs dans son environnement. Si l'amélioration est notable par rapport à ATEMU au niveau de l'étude du processeur, il n'apporte pas de solutions aux carences d'ATEMU, identifiées précédemment.

2.4 Synthèse

Le nombre important de simulateurs nécessite une analyse et un comparatif. En effet, chacun présente des fonctionnalités intéressantes, offrant la possibilité d'étudier le comportement d'un

réseau de capteurs mais souvent de manière partielle ou fortement dépendante du matériel. Nous recherchions quatre fonctionnalités importantes dans les approches existantes :

1. **Environnement** : le modèle doit intégrer la possibilité de générer des scénarii environnementaux ; dans le cadre de notre étude, nous désirons pouvoir étudier un scénario de feux de forêt.
2. **Modèle énergétique** : l'énergie, aspect crucial dans la durée de vie des capteurs, doit pouvoir être représenté de manière précise.
3. **Composants matériels** : le modèle doit pouvoir prendre en compte tous les composants d'un capteur et avoir la capacité d'étudier le comportement de chacun.
4. **Topologie ou déploiement** : le modèle doit favoriser l'étude du déploiement du réseau et les obstacles du monde réél doivent pouvoir être pris en compte.

   Toutes les approches étudiées sont toutes incomplètes étant donné les fonctionnalités recherchées. Cependant, l'étude que nous venons de réaliser, nous a permis d'indentifiés d'autres fonctionnalités :

5. **Personnalisation** : le modèle doit autoriser les modifications dans le cadre de recherche, en particulier dans la possibilité de travailler sur l'implémentation rapide et simplifiée de plusieurs protocoles de routage par le biais d'un outil dédié comme semble le proposer l'outil Wirokit de l'approche SWAN.
6. **Modularité** : le modèle doit être décomposable en sous-parties à l'image de la librairie d'outils tel d'ATEMU.
7. **Généricité** : le modèle doit être indépendant du matériel, c'est à dire qu'il ne doit pas prendre en compte des particularités de constructeur mais reproduire le comportement d'un composant dans son aspect global, matériel indépendant comme l'approche SENS.

Nous proposons un comparatif des simulateurs de réseaux de capteurs sans fil illustré par le Tableau 2.2 en fonction des critères recherchées cités ci-dessus. Les critères que nous avons formulés sont décisifs pour répondre à l'objectif de notre étude. Nous notons par un + et par - chaque critère, avec le doublement du signe + + si ce critère est un point fort de l'approche ou bien - - si

le critère fait cruellement défaut. Nous notons par "floue" le cas où une fonctionnalité n'est pas clairement définie.

	Matériels	**Personnalisation**	**Modularité**	**Généricité**	**Environnement**	**Topologie**	**Energie**
TOSSIM/PowerTossim	+ +	-	-	- -	-	- -	+
SensorSim	-	+	+	+	-	+	+
SENS	-	+	+ +	+ +	+	floue	-
SWAN	-	+	- -	- -	+	-	- -
Glonemo	+	-	-	-	++	-	- -
ATEMU	-	-	+ +	- -	-	-	- -
Avrora	+	-	-	-	-	-	+ +

TAB. 2.2*: Comparaison des simulateurs*

Nous ne trouvons pas, dans l'inventaire réalisé sur les différentes approches, le moyen d'étudier un réseau de capteurs dans la problématique des feux de forêt. Il nous apparaît donc nécessaire de créer un outil en respectant les fonctionnalités indentifées précédemment, définissant un cadre de conception.

Une définition complète de tous les composants, la possibilité de pouvoir étudier différents phénomènes environnementaux, d'analyser la consommation énergétique doivent être offerte par l'outil. De plus cet utilitaire, doit être facilement modifiable, modulaire et générique.

Pour modéliser un réseau de capteurs sans fil, il est possible d'utiliser différents formalismes. De nombreux travaux ont utilisé les réseaux de Petri [Jiao et al., 2005], les chaînes de Markov

[Achir et Ouvry, 2005][Chiasserini et Garetto, 2004b], les réseaux bayésiens[Jiao et al., 2005] ou le formalisme DEVS [Farooq et al., 2004]. Mais souvent ces travaux sont incomplets car ils ne présentent que de manière partielle le comportement d'un réseau. De manière générale, ils traîtent le routage et la consommation énergétique. Il n'y a pas de réelles définitions des composants d'un capteur sans fil ou d'études sur les interactions environnementales. Ces aspects sont importants pour étudier le comportement fonctionnel d'un capteur. Cependant nous allons nous intéresser plus particulièrement au formalisme DEVS car c'est un formalisme très élaboré

et dont le laboratoire SPE a l'expérience. De plus nous avions pu découvrir et apprécier les capacités de DEVS dans le cadre de travaux de modélisation et de simulation en bioinformatique [Antoine-Santoni et al., 2007a].

Quelques travaux fournissent une approche sur la modélisation et la simulation des protocoles de routage dans les réseaux sans fil et les réseaux de capteurs, en introduisant la théorie des systèmes par le biais du formalisme DEVS. Le formalisme DEVS (Discrete EVent system Specification) a été introduit par le professeur B.P. Zeigler vers la fin des années 70 [Zeigler, 1976, Zeigler et al., 2000] et certains travaux ont commencé à utiliser ce formalisme pour décrire le comportement des réseaux sans fil. Les travaux de [Farooq et al., 2004] proposent la modélisation et la simulation de l'algorithme de routage AODV (Ad-hoc On Demand Vector). Ils proposent de modéliser et de simuler ce protocole de routage à l'aide du formalisme Cell-DEVS, formalisme utilisant des cellules définies comme des modèles DEVS, dans le cas des réseaux *ad-hoc*. Si ce travail prouve la capacité d'utiliser ce formalisme pour visualiser des performances de routage, il ne s'inscrit en aucune manière dans la définition des réseaux de capteurs sans fil. Des travaux plus récents de [Kim, 2006b] proposent l'intégration d'une application DEVS dans le simulateur de réseau très populaire NS-2. Dans le cadre d'une application de détection et de destruction de chars (contexte d'application militaire), les auteurs proposent d'intégrer un modèle DEVS pour la définition des composants d'un réseau de capteurs. L'objectif est de fournir au simulateur NS-2, qui est un outil complexe dans sa définition et dans son utilisation comme le précise [Kim, 2006b], une partie DEVS pour la définition de réseaux de capteurs dans le cas applicatif d'un champ de bataille. Les auteurs utilisent le formalisme DEVS dans cette application en considérant les avantages du formalisme DEVS, tels que la modularité, la réusabilité et la propriété de pouvoir fournir un simulateur à chaque modèle créé. Cependant ces travaux ne proposent pas de modéliser en DEVS un capteur dans sa totalité, et ne permettent non plus l'implémentation rapide des caractéristiques des noeuds dans la communication (protocole de routage), fortement dépendante du simulateur complexe NS-2. L'approche d'un modèle environnemental n'est pas réellement proposée et cela ne correspond pas à nos attentes dans la problématique des feux de forêt. Cependant dans tous ces travaux, le formalisme DEVS est défini comme idéal pour décrire la nature asyn-

chrone des événements se déroulant dans les réseaux de capteurs sans fil. Le formalisme DEVS dans l'approche DEVS-NS2 est utilisé pour sa modularité, sa réusabilité et la propriété de pouvoir fournir automatiquement un simulateur à chaque modèle créé. Ces avantages représentent une base sérieuse dans notre réflexion sur définition d'un nouvel outil de modélisation et de simulation d'un réseau de capteurs sans fil.

2.5 Conclusion

Dans ce chapitre, nous avons défini les principaux termes et concepts propres au domaine des réseaux de capteurs sans fil. Les composants matériels essentiels et la norme *Zigbee* ont été présentés. Nous avons ensuite focalisé notre étude sur la notion de protocole de routage, en identifiant les caractéristiques idéales dans la problématique des feux de forêt, aboutissant au choix de la famille du routage dit "à plat". Dans l'optique de l'étude des protocoles de routage, nous avons présenté les différentes approches de modélisation et de simulation dans les réseaux de capteurs sans fil. Cela nous a permis d'affirmer que les nombreuses approches existantes n'étaient pas complètes dans leur définition et qu'il était nécessaire de proposer un nouvel outil. Cet outil doit fournir une description fine des composants, analyser différentes stratégies de déploiement dans un scénario de feu de forêt. DEVS par sa modularité, sa réusabilité s'affirme comme une approche efficace de modélisation et de simulation. Nous proposons de voir dans la partie suivante les fondements qui vont conduire à la modélisation et à la simulation d'un nouvel outil de modélisation et de simulation de réseaux de capteurs sans fil, basé sur le formalisme DEVS.

CHAPITRE 3

Fondements de notre approche

Le chemin est long du projet à la chose.

Molière

Ce chapitre introduit les fondements de notre approche. La première partie présente les concepts essentiels de modélisation et de simulation avec les notions de système, de modèle, de modélisation et de simulation. Dans une seconde partie, nous nous attachons à définir le formalisme DEVS dans l'aspect modélisation et simulation sur lequel se base notre réflexion. Enfin une conclusion vient synthétiser les points importants de ce chapitre.

3.1 Concept de la modélisation et de la simulation

Cette partie introduit les notions essentielles de la théorie de la modélisation et de la simulation de systèmes complexes. La modélisation est l'établissement d'un modèle pour répondre à des questions et la simulation peut se définir comme l'étude de l'état d'un modèle à travers le temps. Le couple modélisation-simulation doit fournir des solutions techniques selon trois objectifs principaux [Bernardi, 2002] :

- la prévision comportementale, qui a pour but l'observation des réactions d'un système face à des excitations externes ;
- le contrôle, qui a pour objectif la gestion d'un système en agissant sur une ou plusieurs de ses composantes ;
- le perfectionnement de systèmes existants ou la conception de nouveaux systèmes dont les conditions de développement sont difficiles à mettre en œuvre (non-observabilité des données, manque de fiabilité, coûts, expériences dangereuses,etc...).

Le processus de modélisation et de simulation fait référence à plusieurs notions qu'il nous semble nécessaire d'éclairer.

Un ***système*** peut être défini comme une combinaison d'éléments, en relation les uns avec les autres et formant un tout [Aiello, 1997]. Cette définition affinée indique qu'un système doit être considéré comme étant un ensemble de sous-systèmes formant une unité, agissant et interagissant pour l'accomplissement d'une fin logique. A partir de cette définition, nous pouvons décrire un système hiérarchiquement, en utilisant des sous-systèmes, considérés eux-mêmes comme des systèmes à part entière. Nous caractérisons un système par sa structure (son architecture interne) et son comportement (ses réactions par rapport aux excitations provenant du milieu extérieur). Ce comportement repose à la fois sur les sollicitations externes et sur l'état (les valeurs des caractéristiques) de ses sous-systèmes internes représentés par des variables.

Le fonctionnement d'un système peut se résumer aux séquences suivantes :

- arrivée d'excitations externes sur les entrées du système ;
- réaction de celui-ci, en tenant compte de son état courant ;
- génération d'une réponse sur les sorties du système.

Un ***modèle*** est *"une description abstraite d'un système ou d'un processus, une représentation simplifiée qui permet de comprendre et de simuler*" pour P.A. Muller [Muller et Gaertner, 2000]. A un plus haut niveau, un modèle peut être vu comme une abstraction de la réalité visant à la mieux comprendre [Muzy, 2004]. Le processus d'identification d'un système et de conception du modèle correspondant est appelé *modélisation*. Selon C. Oussalah, *"la* modélisation *est un épimorphisme entre un système réel et un modèle, dont la finalité est de donner une représentation simplifiée*

et observable de la structure et du comportement du système réel" [Oussalah, 1999] . Pour P.A Fishwick [Fishwick, 1995], "*modéliser est abstraire de la réalité une description d'un système dynamique*".

Ces modèles sont couplés avec une structure de contrôle appelée ***simulateur***, de manière à produire un comportement possible du système dans certaines conditions. Ce processus de couplage est appelé *simulation*. J. Popper définit comme le couple modélisation-simulation *"l'ensemble des activités qui consistent à mettre au point, construire, tester, valider, utiliser et analyser un modèle formel élaboré pour représenter les aspects d'un système retenus comme significatifs pour les objectifs de l'étude*" [Popper, 1973].

Un processus de modélisation et de simulation mettra en œuvre trois entités principales (figure 3.1) :

- le *système* (réel ou fictif) fournissant les données comportementales nécessaires à l'élaboration du modèle et à la vérification des résultats ;
- le *modèle* servant de support de raisonnement lors de l'étude ;
- le *simulateur* gérant le modèle afin de produire des données comportementales qui seront comparées aux informations du système.

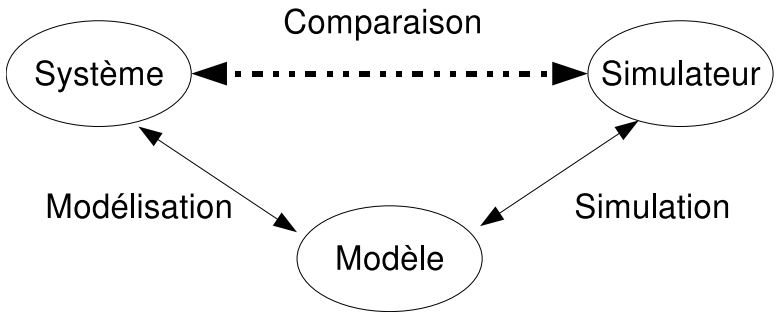


FIG. 3.1*: Les entités du processus de modélisation et de simulation*

Ces trois entités sont mises en relation grâce à deux liens. Le premier d'entre eux est le lien de modélisation qui relie le système et le modèle. Il symbolise le moyen de compiler les connaissances du système réel. Le second est le lien de simulation qui relie le modèle et le simulateur.

Il représente quant à lui l'échange d'instructions permettant la génération des résultats. La comparaison des résultats issus du simulateur et de ceux issus du système étudié vont autoriser la validation du modèle si les critères de formulation sont validés.

De par leur nature même de représentation d'un système, les modèles sont développés selon une certaine *perspective* propre à leur concepteur et au domaine auxquels ils appartiennent. Cette perspective est en fait un mécanisme qui permet au concepteur de déterminer quelles sont les informations pertinentes à l'accomplissement de la modélisation qu'il est en train d'effectuer comme le clarifie [Bernardi, 2002] . Le *niveau d'abstraction* d'un modèle est une notion corollaire et dépendante de la perspective qui détermine la quantité d'informations contenue dans celui-ci. Cette notion permet de ne considérer que les informations pertinentes à la modélisation en considérant les abstractions comme un moyen de cacher sélectivement celles qui ne se révèlent pas indispensables [Benjamin et al., 1993]. Nous pouvons noter que la quantité d'information diminue avec le niveau : un modèle décrit à un bas niveau d'abstraction contiendra plus d'informations qu'un modèle décrit à un niveau d'abstraction supérieur. Un modèle décrit selon différents niveaux d'abstraction est appelé modèle hiérarchisé [Fishwick, 1998]. Le choix d'un niveau d'abstraction est une étape importante dans la conception d'un modèle et est le plus souvent effectué très tôt dans le processus de modélisation et de simulation. La détermination du niveau d'abstraction le plus approprié dès les premières étapes du processus fait l'objet de nombreuses références dans la littérature. Il s'agit donc de sélectionner le niveau d'abstraction en fonction des résultats que l'on est en droit d'attendre ou des données dont on dispose[Oussalah, 1998, C. Oussalah et Thalens, 1995].

Le processus de modélisation et de simulation décrit précédemment est un processus général adapté à tous les types de modélisation et de simulation. Dans [Stein et Loucas, 1995], les auteurs définissent un processus de modélisation se décomposant en trois étapes : la *décomposition*, la *synthèse* et la *déduction du modèle*.

La première étape, celle de *décomposition*, consiste à isoler le modèle de son environnement (en déterminant les entrées/sorties) et à identifier les composants du système à modéliser. La seconde étape, l'étape de *synthèse*, consiste à créer et à sélectionner des modèles de composants (appelés « templates »), à les interconnecter pour former le modèle complet et à définir des

équations d'état. Enfin, la dernière étape, l'étape de *déduction du modèle*, consiste à analyser le modèle obtenu et à utiliser un algorithme pour déterminer s'il est « convenable » (« Proper Model »), c'est-à-dire s'il est conforme aux attentes du constructeur du modèle.

Nous baserons notre réflexion sur cette approche en utilisant pour cela un formalisme à événements discrets, issu de la théorie des systèmes.

3.2 Le formalisme DEVS

Basé sur la théorie des systèmes, le formalisme DEVS (Discrete EVent system Specification) a été introduit par le professeur B.P. Zeigler vers la fin des années 70 [Zeigler, 1976]. Il permet une modélisation modulaire et hiérarchique des systèmes à événements discrets. Dans le formalisme DEVS un modèle est vu comme une "boîte noire" qui reçoit et émet des messages sur ses ports d'entrées ou de sorties. Cette section introduit le formalisme DEVS en distinguant la partie modélisation de la partie simulation. Dans la première partie nous mettons en évidence les concepts de modularité et de hiérarchie en présentant deux types de modèles : les modèles atomiques et les modèles couplés. Dans la seconde partie nous montrons comment il est possible, à partir des modèles, de générer automatiquement les algorithmes de simulation associés. Nous verrons plus particulièrement comment à partir d'une notion de messages il est possible de mettre en oeuvre ces algorithmes.

3.2.1 La modélisation

Le formalisme DEVS définit deux catégories de modèles : les modèles atomiques et les modèles couplés. Les modèles atomiques sont chargés de représenter le(s) comportement(s) du système. Les modèles couplés sont définis par un ensemble de sous-modèles (atomiques et/ou couplés) et traduisent la structure interne des sous-parties du système grâce à la définition de couplage entre les sous-modèles.

3.2.1.1 La notion de modèle atomique

Le formalisme de base DEVS considère un modèle atomique comme un concept mathématique basé sur le temps, un ensemble de valeurs caractérisant tous les stimuli possibles en entrée et en sortie du système ainsi que deux fonctions permettant de déterminer la réponse comportementale à ces stimuli. Dans le formalisme DEVS, la notion de couple (port, valeur) est introduite pour chaque port (entrée ou sortie) d'un modèle atomique. Les spécifications classiques d'un modèle atomique MA avec ports sont les suivantes :

$$MA =< X, Y, S, \delta_{int}, \delta_{ext}, \lambda, t_a >$$

où,

- X = {(p, v) | p ∈ Ports_entrée, v ∈X_p } est la liste des ports et des valeurs d'entrées,
- Y = {(p, v) | p ∈ Ports_sortie, v ∈ Y_p } est la liste des ports et des valeurs de sorties,
- $t_a : S \in \mathbb{R}+$ est le temps de vie de l'état S ,
- S est l'ensemble des variables d'états,
- $\delta_{int} : S \rightarrow S$ est la fonction de transition interne,
- δ_{ext} : Q × X →S est la fonction de transition externe où,
  - Q = {(s, e) | s ∈ S, 0 ≤ e ≤t_a (s)} est l'ensemble des états,
  - e est le temps écoulé depuis la dernière transition.
- $\lambda : S \rightarrow Y$ est la fonction de sortie,.

Les modèles réagissent à deux types d'événements : les événements externes et les événements internes. Les événements externes proviennent d'un autre modèle, déclenchent la fonction de transition externe et mettent à jour le temps de vie du composant. Les événements internes correspondent à des changements d'états du modèle, et déclenchent les fonctions de transitions internes et de sorties. Le modèle calcul ensuite, grâce à la fonction t_a, la date du prochain événement interne.

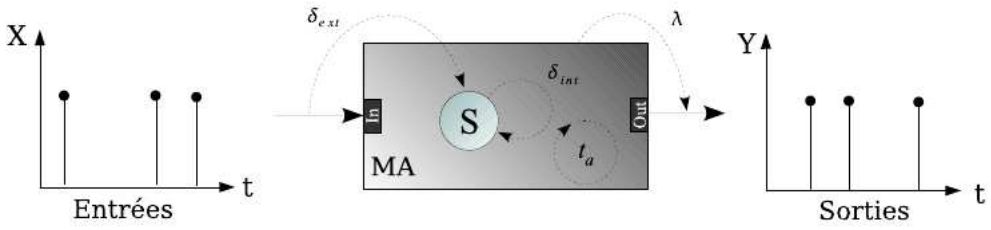


FIG. 3.2*: Fonctionnement d'un modèle atomique*

L'interprétation des ces éléments est illustrée sur la Figure 3.2. A chaque instant le système, le modèle atomique (MA) est dans un état s. Si aucun événement externe n'intervient, le système restera dans cet état pendant un temps donné par la fonction t_a (s). Lorsque le temps de vie du modèle est expiré, c'est à dire lorsque qu'il s'est écoulé e = t_a (s) le système active sa fonction de sortie λ(s) et change d'état grâce à l'exécution de la fonction de transition interne δ_{int} (s). Si un événement externe x $\in X$ intervient avant que le temps ne soit expiré, c'est à dire quand le système est dans l'état total (s, e) avec e$\leq t_a$ (s), le système change d'état grâce à l'exécution de la fonction de transition externe δ_{ext} (s, e, x). Dans les deux cas, le système est alors dans un nouvel état s avec un nouveau temps restant t_a (s) et ainsi de suite. Le temps de vie du composant peut être égal à zéro ou à l'infini. Dans le premier cas, la durée de l'état s est tellement courte qu'aucun événement externe ne peut intervenir avant l'arrivée du prochain changement d'état. Nous disons de s qu'il est dans un état transitoire. Dans le second cas, le système restera dans l'état s indéfiniment si aucun événement externe ne vient l'interrompre. Nous disons dans ce cas que s est un état passif.

3.2.1.2 Le modéle couplé

Le formalisme DEVS utilise la notion de hiérarchie de description qui permet la construction de modèles dits "couplés" à partir d'un ensemble de sous-modèles et de trois relations de couplage avec ces sous-modèles. Un modèle couplé est constitué de sous-modèles qui peuvent être atomiques ou couplés et il possède les trois relations de couplage suivantes comme illustré sur la Figure 3.3 :

– une relation de couplage interne pour le couplage entre les ports des sous-modèles qui composent le modèle couplé ;
– une relation de couplage des entrées externes pour le couplage entre les ports d'entrées du modèle couplé et les ports d'entrées des sous-modèles ;
– une relation de couplage des sorties externes pour le couplage entre les ports de sorties du modèle couplé et les ports de sorties des sous-modèles.

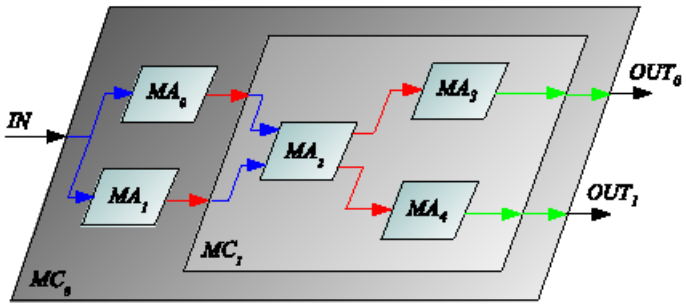


FIG. 3.3*: Modèle couplé : définition des couplage*

La Figure 3.3 montre un exemple de hiérarchie entre les sous-modèles d'un système. Le modèle couplé MC_0 modélise au plus haut niveau le système étudié. Il possède deux ports de sortie OUT_0 et OUT_1, un port d'entrée IN et il contient les deux modèles atomiques MA_0 et MA_1 ainsi qu'un modèle couplé supplémentaire MC_1. Les modèles atomiques MA_0 et MA_1 sont reliés par une relation de couplage d'entrée externe au modèle couplé MC_0, et par une relation de couplage interne au modèle couplé MC_1. Dans le cas du formalisme DEVS classique avec port les spécifications d'un modèle couplé sont les suivantes :

$$MC = < X, Y, D, Mi, Ii, Zi, j >$$

où,

– X = {(p, v) | p ∈ Ports_entrée, v ∈X_p } est la liste des ports et des valeurs d'entrées,

- Y = {(p, v) | p ∈ Ports_sortie, v ∈ Y_p } est la liste des ports et des valeurs de sorties,
- D est la liste des composants constituant le modèle couplé ,
- $Mi =< Xi, Yi, Si, \delta_{int,i}, \delta_{ext,i}, \lambda i, t_{a,i} >$ est un modèle atomique,
- Pour chaque modèle $i \in D \cup \{MC\}$, I_i est l'ensemble des modèles qui influencent i,
- $Z_{i,j}$ est la fonction de translation des sorties de i vers j telles que :
- $Z_{MC,j} : X_{MC} \rightarrow X_j$ est la fonction de couplage des entrées externes,
- $Z_{i,MC} : Y_i \rightarrow X_{MC}$ est la fonction de couplage des sorties externes,
- $Z_{i,j} : Y_i \rightarrow X_j$ est la fonction de couplage interne.

La structure d'un modèle couplé doit répondre à des contraintes telle que,

1. $Mi =< Xi, Yi, Si, \delta_{int,i}, \delta_{ext,i}, \lambda i, t_{a,i} >$ est un modèle atomique,
2. une seule fonction $Z_{i,j}$contient l'ensemble des informations sur les couplages du modèle couplé,
3. $I_i$est un sous-ensemble de $D \cup \{MC\}$ et $i \notin I_i$.

La cohérence et la conservation des comportements du système entre ces niveaux hiérarchiques est résumée par la propriété dite "closed under coupling" [Zeigler, 1990]. Cette propriété assure qu'un modèle couplé, représenté par le couplage d'un ensemble de sous-modèles plus détaillés, est équivalent à un modèle atomique.

3.2.2 La simulation

L'une des propriétés importantes du formalisme DEVS est qu'il fournit automatiquement un simulateur pour chacun des modèles. DEVS établit une distinction entre la modélisation et la simulation d'un système tel que n'importe quel modèle DEVS puisse être simulé sans qu'il soit nécessaire d'implémenter un simulateur spécifique. Chaque modèle atomique est associé à un simulateur chargé de gérer le comportement du composant et chaque modèle couplé est associé à un coordinateur chargé de la synchronisation temporelle des composants sous-jacents. L'ensemble de ces coordinateurs et simulateurs est géré par un coordinateur spécifique appelé "Root".

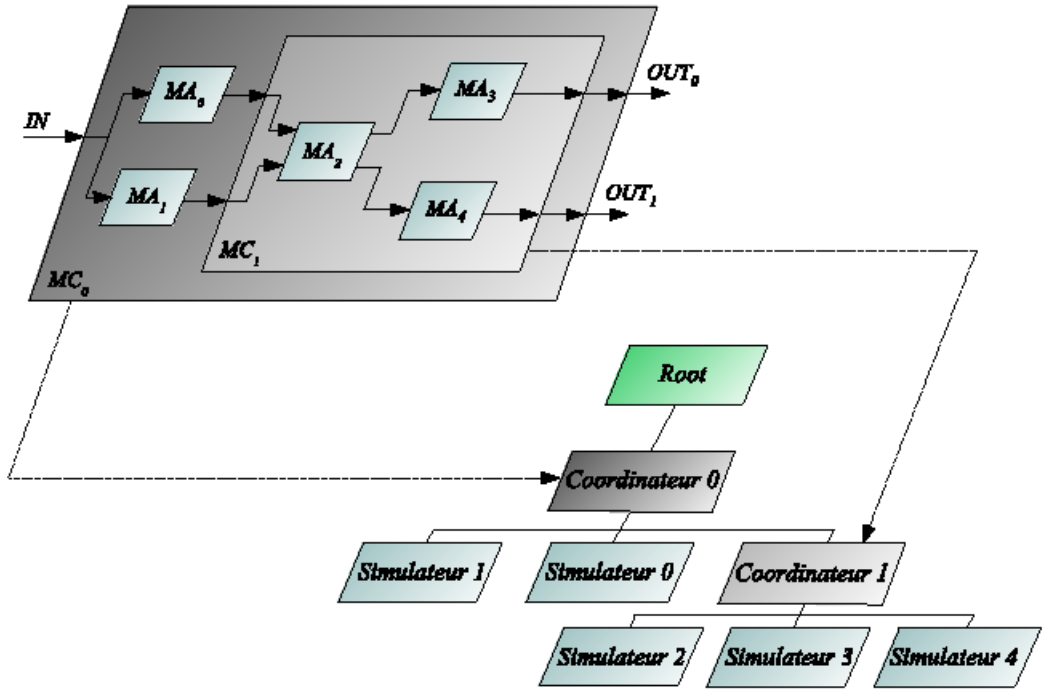


FIG. 3.4*: Arbre hiérarchique de la simulation DEVS*

La hiérarchie du simulateur DEVS est construite sur une arborescence constituée de simulateurs et de coordinateurs comme illustré sur la Figure 3.4. Les deux coordinateurs “Coordinateur 0” et “Coordinateur 1” sont associés aux modèles couplés MC_0 et MC_1. Le “Coordinateur 0” est chargé de gérer le “Coordinateur 1” ainsi que les simulateurs “Simulateur 0” et “Simulateur 1” associés aux deux modèles atomiques MA_0 et MA_1. De la même manière, comme le modèle couplé MC_1 encapsule les trois modèles atomiques MA_2 , MA_3 et MA_4, le coordinateur “Coordinateur 1” associé gère les trois simulateurs “Simulateur 2”, “Simulateur 3” et “Simulateur 4”. Le coordinateur “Root” est chargé de coordonner la totalité des composants de l’arbre de simulation. L’ordre d’exécution des fonctions comportementales (δ_{int}, δ_{ext},λ, t_a) d’un modèle atomique DEVS sous-entend la présence d’un mécanisme de simulation. Ce mécanisme est géré par le composant simulateur qui, comme le coordinateur, peut recevoir ou émettre 4 types de messages :

– le message d’initialisation (i,t) permet à tous les acteurs d’effectuer une synchronisation temporelle initiale ;
– le message de transition interne (*,t) permet la gestion d’un événement interne avec l’exécution de la fonction δ_{int} ;

- le message de sortie (y,t) permet le transport des sorties (données par λ) aux éléments parents et résulte de la réception d'un message (*,t) ;
- le message de transition externe (x,t) permet la gestion d'un événement externe avec l'exécution de la fonction δ_{ext} .

Le formalisme DEVS est utilisé pour la description de systèmes à événements discrets. Il constitue un outil de modélisation et de simulation permettant de représenter de manière modulaire et hiérarchique les systèmes réagissant à des événements discrets et de les simuler.

3.3 Conclusion

La méthode qui consiste à décomposer, synthétiser, déduire un modèle sera utilisée dans le cadre de modélisation d'un réseau de capteurs sans fil. Pour représenter ce système, nous utiliserons le formalisme DEVS. En effet, il possède certains avantages que nous énumérons : il permet la modélisation d'un système sur plusieurs niveaux de description ; il permet la distinction entre la modélisation et la simulation d'un système ; il permet la simulation des systèmes automatiquement à partir de ces modèles ; il permet de simuler des modèles continus ; il rend facile la modification du comportement des modèles. Ces nombreux aspects nous orientent vers l'utilisation du formalisme DEVS dans la définition de notre approche de modélisation et de simulation. Dans le chapitre suivant, nous présentons l'application DEVS-Wireless Sensor Network : **DEVS-WSN** développé dans le but de permettre l'étude des réseaux de capteurs sans fil dans la problématique d'étude de phénomènes environnementaux.

CHAPITRE 4

Modélisation d'un capteur sans fil

Une réalité complexe reste toujours inachevée et incomplète. On ne peut jamais la comprendre ou la maîtriser entièrement ou définitivement. Dès qu'on pense avoir trouvé des schémas explicatifs globaux, ils se modifient, laissant place à d'autres modèles tout aussi éphémères.

extrait de "Manager la complexité", *Dominique Génelot*

Nous proposons dans ce chapitre une application pour la modélisation et la simulation de réseaux de capteurs sans fil. Dans une application des réseaux de capteur sans fil, l'observateur est intéressé par la surveillance d'un phénomène. Nous considérons un réseau comme un ensemble d'entités échangeant des informations, soumis à différentes contraintes physiques et matérielles, dans le but d'accomplir une tâche (surveillance environnementale). L'objectif ultime du réseau est de collecter des informations environnementales et de les transmettre à l'utilisateur. L'outil que nous proposons se base sur ce principe mais il est également nécessaire de rappeler que d'autres axes vont conduire à la modélisation du système :

– les capteurs sont composés de différents éléments matériels (*cf* 2.1.1) ; la définition d'un capteur présenté par [Akyildiz et al., 2002c] regroupe cinq composants principaux : outil de

communication, source d'énergie, capteurs environnementaux, processeur et mémoire ;

– les entités du réseau échangent des messages ; il s'agit de déterminer une structure de message qui sera échangée entre les noeuds ;
– ces éléments possèdent des règles de communication, identifiées sous le terme de protocole. (*cf* 2.1.2). Il est nécessaire de prendre en compte le protocole de routage mais également le protocole d'accés média (MAC). Nous rappellons que le protocole MAC ne constitue pas notre axe principal de recherche.

L'outil, que nous proposons, va permettre de visualiser le système selon trois angles de vue : la vue composant (description des comportements de chaque modèle de composant), la vue du capteur (échange de messages entre les noeuds) et la vue du réseau dans sa globalité (déploiement). Le développement de cette application se fait dans un souci de généricité, de modularité et de "modifiabilité" ou personnalisation, fonctionnalités que nous avons énoncé dans le Chapitre 3.

Dans une première partie, nous détaillons notre modèle original de capteurs basé sur une description complète de tous les composants matériels. La définition régissant les communications intra et inter-noeuds du réseau est décrite dans la seconde partie avec la prise en compte du protocole MAC. Nous proposons dans une troisième partie, une traduction algorithmique de deux techniques de routage issues de la famille des protocoles "à plat" et la proposition d'un nouvel algorithme de routage Variant Of Xmesh (VOX). Enfin la partie finale représente la conclusion de ce chapitre.

4.1 Approche de la description dynamique

4.1.1 Structure des messages

Le message est l'élément qui va permettre le transfert d'informations entre les entités du réseau. Nous présentons sur la Figure 4.1, la structure du message.

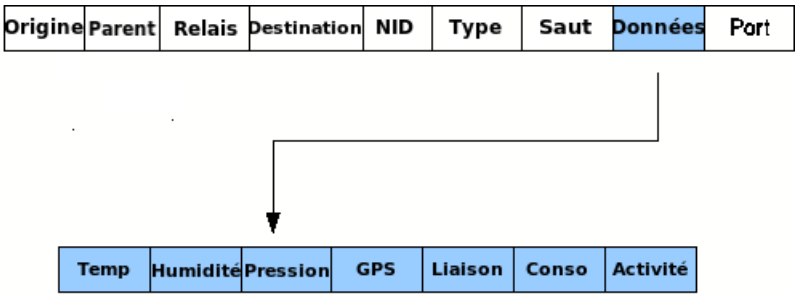


FIG. 4.1*: Structure du message*

Différents champs constituent la trame du message et ces champs jouent un rôle particulier à chaque étape de l'analyse d'un message.

Nous énumérons ces champs en détaillant leur correspondance :

- ***Origine*** : ce champ indique le nom de l'entité génératrice du message ;
- ***Relais*** : il détermine l'entité émettrice du message, qui est souvent un relais entre l'origine et la destination finale ;
- ***Destination*** : il indique le nom du noeud de destination lors de l'émission d'un message ;
- ***NID*** : il détermine le champ d'identification du groupe auquel le noeud peut appartenir ;
- ***Type*** : il précise l'objectif du message. Il indique l'action que devra réaliser le processeur à la lecture de ce message ; il existe différents types que nous détaillons plus loin mais par exemple nois citons le type *WhiteFlag,* message d'initialisation des communications ;
- ***Saut*** : il fournit une indication sur le nombre de sauts réalisés avant son arrivée ;
- ***Données*** : regroupant plusieurs champs, il contient toutes les données environnementales, de localisation GPS, d'activité du processeur (activité) et de capacité énergétique (conso) du noeud qui génère un message ;
- ***Port*** : propre à notre approche, il va indiquer sur quel port de sortie doit s'orienter le message.

Nous faisons une distinction entre une entité émettrice et une entité génératrice par les champs ***Relais*** et ***Origine***. En effet un noeud peut émettre un message mais aura besoin de relais pour transmettre l'information dans le cas de communication indirecte entre le noeud et la station de base. Tous les noeuds qui recevront l'information verront le champ ***Relais*** et modifieront ce champ

au fur et à mesure de la communication, mais le champ ***Origine*** ne sera pas dénaturé, pour autoriser la station de base à visualiser, par l'intermédiaire de cette donnée, le noeud générateur du message. Après avoir vu la structure d'un message, nous allons nous attacher à détailler notre modèle de capteur.

4.1.2 Le modèle général de capteur

Nous présentons à présent le Modèle Couplé (MC) représentant les couplages entres les différents composants, modèle essentiel dans notre approche[Antoine-Santoni et al., 2007c]. Nous distinguons à l'intérieur de ce MC "*Sensor*", les différents composants essentiels. Ce MC général se présente comme le couplage de 5 Modèles Atomiques (MA) , "*COM*", "*Memory*","*Battery*", "*SensorBoard*" et "*Env*" et d'un MC nommé "*Process*".

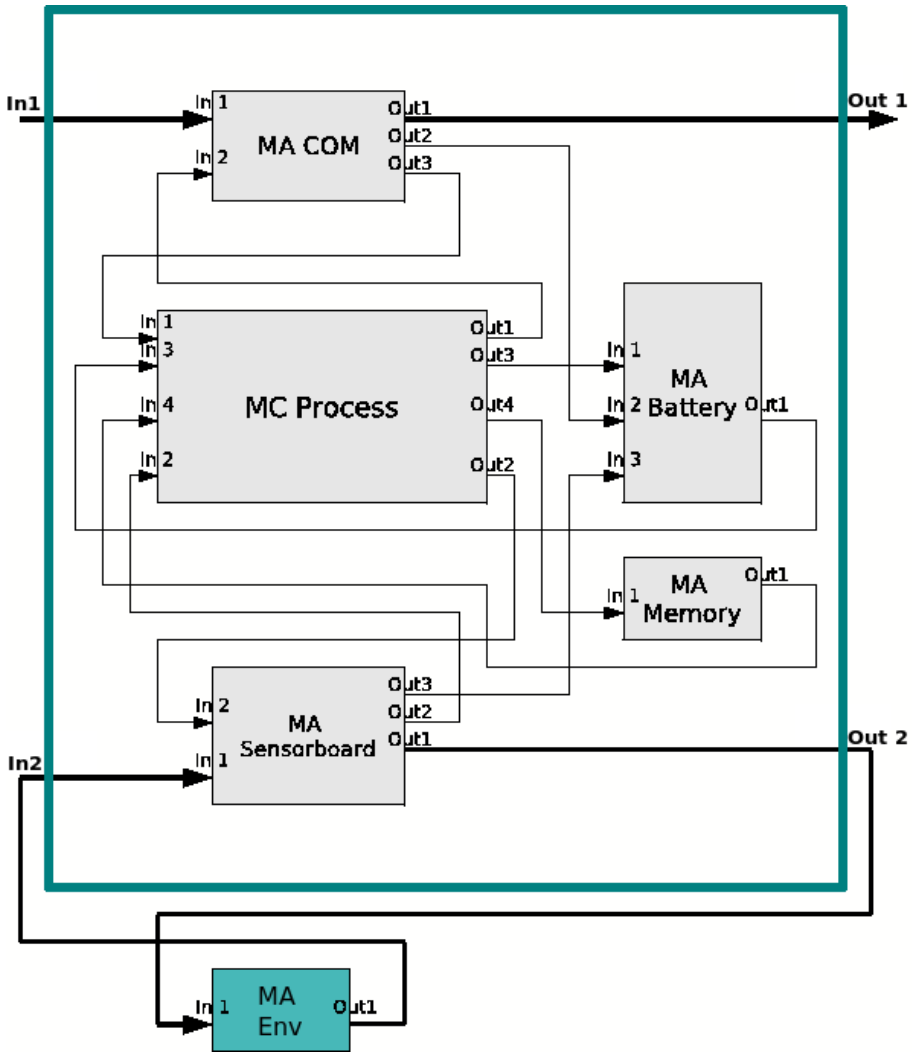


FIG. 4.2*: Notre modèle MC DEVS* "Sensor"

Nous allons détailler à présent chacun des modèles qui constituent le MC "*Sensor*".

4.1.3 Le modèle de communication

4.1.3.1 Description

L'outil de communication a pour but de représenter le dispositif qui va permettre à l'entité de communiquer avec les autres éléments du réseau. Cette communication se fait le plus généralement par le biais d'une antenne radio. Les ondes radio subissent selon l'environnement des atténuations du signal. Notre objectif ici n'est pas de représenter un modèle précis de traitement

du signal radio, mais plutôt d'instaurer une base de communication pour autoriser les échanges d'informations entre les capteurs. Pour cela nous créons un MA DEVS nommé *"COM"* qui représente l'antenne, illustré par la Figure 4.3

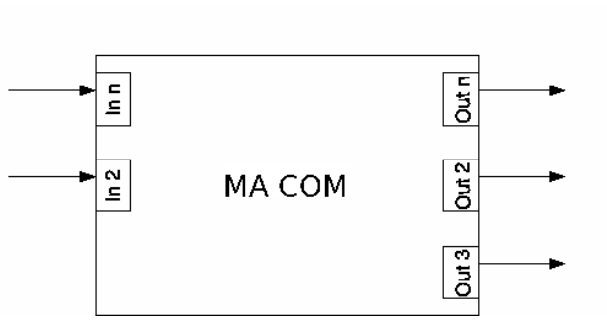


FIG. 4.3*: MA DEVS "*COM*"*

L'objectif de ce modèle est double et réagit selon deux processus de communication : interne et externe. Premièrement, il doit réceptionner les messages externes provenant des autres noeuds et les envoyer à l'unité de traitement interne de l'information. Deuxièmement, il doit réceptionner les messages provenant des processus internes au noeud et envoyer le message vers le bon destinataire dans le cas des communications externes entre les noeuds. A cela s'ajoute une fonction de consommation énergétique se traduisant par l'envoi d'un message de consommation d'énergie vers le MA "*Battery*".

Communcation externe au noeud : Ce modèle atomique présente plusieurs ports d'entrée de communications externes. Un port d'entrée nommé In n permet les communications avec les autres noeuds. En effet, sur la Figure 4.3, nous ne précisons qu'un seul port d'entrée noté In n par souci de lisibilité du schéma. Un noeud peut avoir plusieurs ports d'entrée en fonction du nombre de noeuds auxquels il sera connecté. Précisons qu'à un port d'entrée de communication externe correspondra un port de sortie de communication externe, que nous notons de la même manière Out n. Il est évident que le noeud a la possibilité de pouvoir communiquer avec autant de noeuds qui seront dans son environnement de communication.

Communication interne au noeud : In 2 représente les informations provenant du processeur et qui seront envoyées à la communication et réciproquement les informations provenant de l'extérieur sont dirigées vers le port de sortie dédié au processeur, Out 2, pour réaliser les procédures de traitement de l'information. Le port de sortie Out 3 correspond à la communication directe vers le modèle énergétique : à chaque action du modèle, une valeur de consommation est envoyée par le biais de port Out 3.

4.1.3.2 Le MA "*COM*"

Nous présentons le diagramme d'état du modéle atomique "*COM*". Nous représentons la dynamique du comportement du MA"*COM*" par l'automate à états finis de la Figure 4.4.

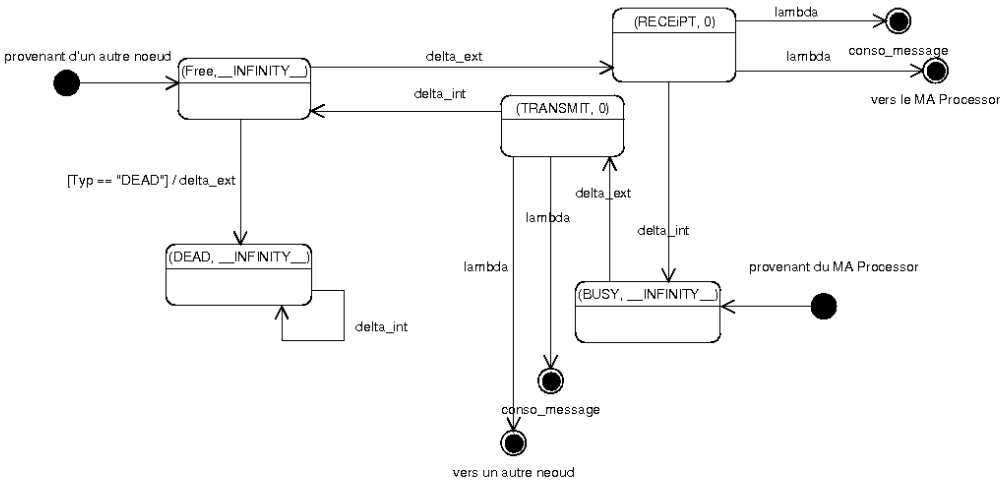


FIG. 4.4*: Graphe d'états du MA "*COM*"*

L'automate présente deux variables d'états que nous présentons :

1. La variable d'état *phase* peut posséder cinq valeurs :
   - *phase* = '***IDLE***' si le modèle est en phase d'attente d'une communication ;
   - *phase* = '***RECEIPT***' si le modèle est en phase de réception d'un message à l'intérieur du modèle ;

- *phase* = '***TRANSMIT***' si le modèle est en phase de transmission d'un message ;
- *phase* = '***BUSY***' si le modèle est en cours de traitement d'un message ;
- *phase* = '***DEAD***' si le modèle est en phase critique et qu'il ne peut plus exécuter d'actions.

2. La variable d'état $sigma \in \mathbb{R}_{\infty}^{+}$ représentant la durée de vie d'un état donné du MA COM.

Si le composant "*COM*" est dans son état initial {'***IDLE***', ∞} et qu'un message est réceptionné sur ses ports de communications externes, la fonction de transition externe δ_{ext} fait passer le modèle dans l'état transitoire de réception { '***RECEIPT***', 0}. Cette phase permet de générer un message vers le port de communication interne. A ce moment-là, le modèle change d'état et se place en phase '***BUSY***', interdisant toute communication. Quand le message a été traité par les autres modèles, il arrive sur le port d'entrée de communication interne. Cet événement déclenche un changement de phase, passant de la phase '***BUSY***' à la phase '***TRANSMIT***'. En phase '***TRANSMIT***', le modèle envoie un message sur ses ports de communication externe. A la fin de cette phase, le modèle passe en phase d'écoute des communications externes '***IDLE***'.

Nous distinguons une phase critique appelée '***DEAD***' . Cet état est la conséquence d'un message arrivant de la communication interne signifiant soit que la ressource énergétique est insuffisante soit le résulat de la détection d'une variable environnementale critique. Cette phase est irréversible et correspond à la "mort" du capteur, rendant ce dernier incapable de réaliser des communications radio.

Pour gérer les communications , nous définissons une table de voisinage qui rescence les différentes connexions. Cette table référence le champ ***Relais*** du message entrant, en créant un annuaire des communications. Une correspondance définit le noeud qui se trouve sur le port d'entrée In1, port In2,...port In n. Ainsi au port d'entrée In1 sera affecté le port de sortie Out1. Cela ne détermine en aucun cas le routage de l'information, mais seulement le voisinage. Le MA "*COM*" aura pour rôle de définir également la qualité des liens durant les communications.

4.1.4 Le modèle de processeur

4.1.4.1 Description

Nous définissons le MC "*Process*". Celui-ci est composé de trois entités distinctes : les MA "*Processor*", "*Net*" et "*Flash*". Nous utilisons le terme *"Processor"*, pour processeur par analogie au rôle central joué par celui-ci au sein des noeuds. Notre modèle a pour but de représenter un comportement fonctionnel d'analyse, de traitement et de modification de certaines informations contenues dans les messages. Le rôle du modèle "*Process*" va être triple :

1. analyser les types d'actions à réaliser ;
2. coordonner les actions des différents composants ;
3. modifier les données réseau du message ;

L'analyse des types d'actions à réaliser va se faire selon le champ ***Type*** du message entrant. En effet, comme décrit précédemment, ce champ va influencer l'orientation du message à l'intérieur des composants du noeud.

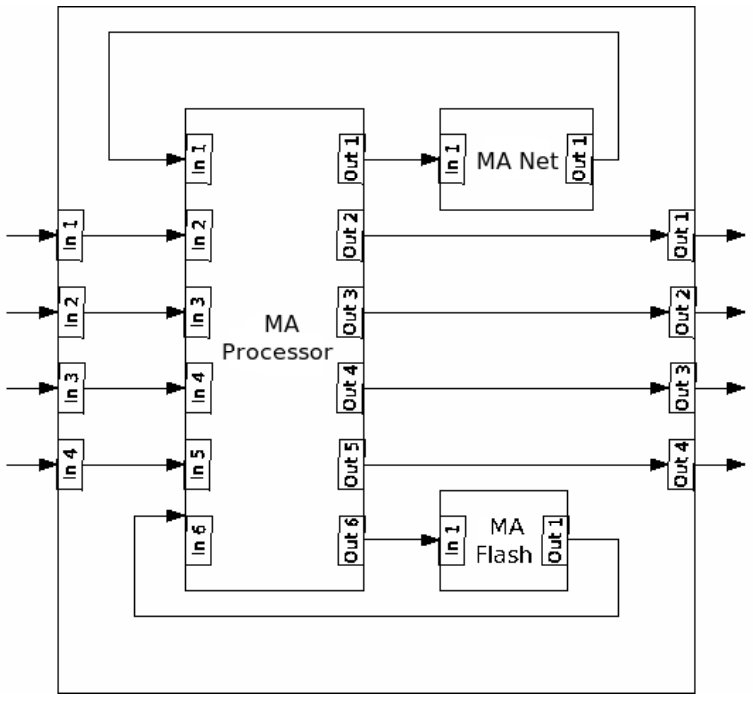


FIG. 4.5*: MC DEVS Processor*

Sur la Figure 4.5, nous distinguons trois MA : "*Processor*", "*Net*" et "*Flash*"

4.1.4.2 Le MA "*Processor*"

Le MA "*Processor*" possède un rôle central et a pour but d'organiser les actions du noeud. Le modèle est connecté à tous les composants du noeud. Il va distinguer les actions à effectuer en fonction de ces ports d'entrée. A chaque port d'entrée va correspondre une liste d'actions. En fonction du champ ***Type***, le message entrant est orienté vers le composant désiré. Le but avoué n'est pas d'avoir une définition précise de l'action du processeur, mais de représenter une structure centrale gérant des opérations basiques pour les répartir au niveau des composants. Ces ports d'entrée sont au nombre de six :

1. le port 1 reçoit des informations provenant du MA "*Net*". Ce dernier a pour but de modifier le champ ***Destination*** selon le protocole de routage ;
2. le port 2 relié au port 1 du MA "*Processor*" reçoit les informations provenant du MA "*COM*". Le champ Type va jouer ici un rôle primordial et déterminer une action à réaliser, c'est à dire une destination vers un composant. Nous présentons dans le Tableau 4.1 les principaux types avec le composant vers lequel il sera orienté par le MA "*Processor*" ;

Type du message	**Destination composant**
WhiteFlag	MA "*Net*"
Collect	MA "*SensorBoard*"
ACK	MA "*Net*"

TAB. 4.1*: Destination du message selon le type arrivant sur le port 2*

3. le port 3 acceuille les données venant du modèle de détection environnementale. A ce niveau, nous distinguons également une orientation différente en fonction du type de message que nous résumons dans le Tableau 4.2 ;

Type du message	**Destination composant**
Stock	MA "*Memory*"
DEAD	MA"*COM*"
Send	MA "Net"

TAB. 4.2*: Destination du message selon le type arrivant sur le port 3*

4. le port 4 recueille le signal provenant du modèle de ressource énergétique. Le port peut recevoir deux types de messages :

   (a) si la batterie du capteur est égale à 0. En effet, seul ce signal critique sera reçu par le modèle processor répercutant l'information jusqu'au modèle atomique pour qu'il passe en phase ***'DEAD'*** ;

   (b) le processeur récupère un message envoyé, complété par la valeur énergétique contenue dans le MA *"Battery"*, information permettant à la station de base d'estimer la capacité énergétique lors de la réception du message ;

5. le port 5 reçoit les données du MA "*Memory*" ;

6. le port 6 receuille les informations du MA "*Flash*".

L'automate d'états finis du MA "*Processor*" possède trois variables d'états :

1. La variable d'état *phase* peut posséder deux valeurs :
   - *phase* = '***FREE***' si le modèle est en phase d'attente d'un message ;
   - *phase* = '***BUSY***' si le modèle est en cours de traitement d'un message ;
2. La variable d'état ***'GO'*** possède 6 valeurs (1,2,3,4,5,6) correspondant aux ports de sortie référencés sur la Figure 4.5.
3. La variable d'état $sigma \in \mathbb{R}_{\infty}^{+}$ représentant la durée de vie d'un état donné du modèle atomique *"COM"*.

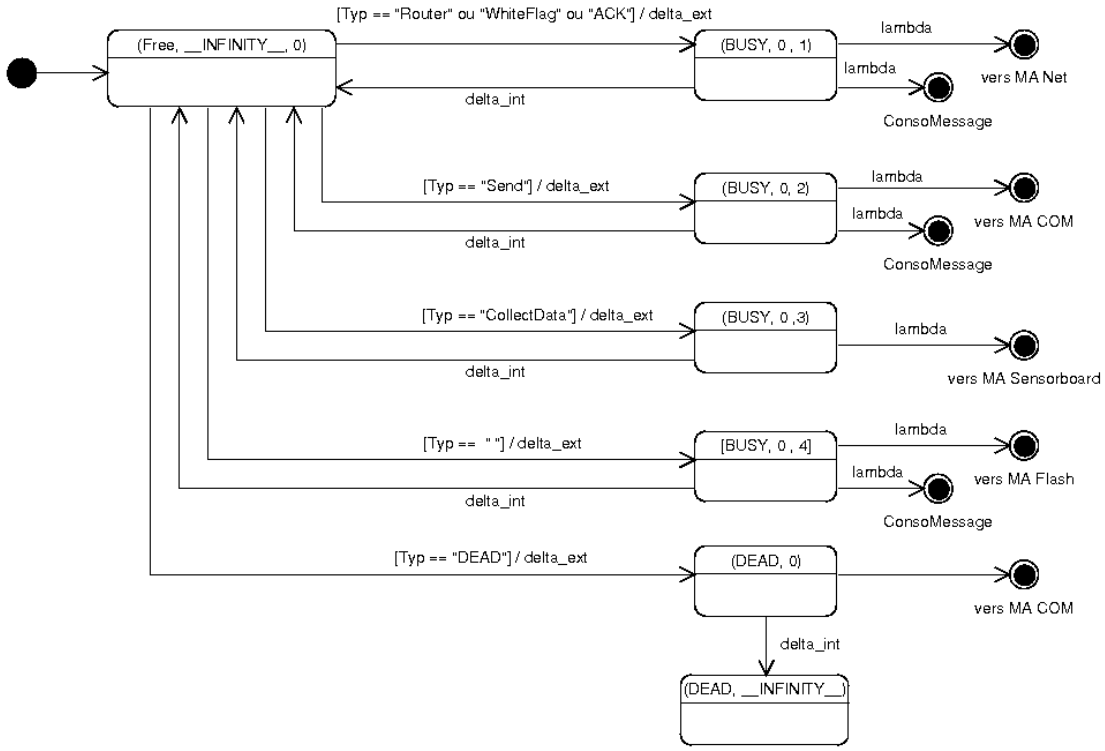


FIG. 4.6*: Graphe d'états du MA* "Processor"

4.1.4.3 Le MA "*Net*"

Le MA "*Net*" est essentiel dans le concept de réseau. Ce modèle aura pour but de fournir les informations de routage pour l'envoi du message. Le routage est fonction du protocole de routage. Notre modèle a un comportement basique. Il possède un port d'entrée In 1 recevant les informations du modèle atomique *"Processor"*. Il possède deux variables d'états :

1. *phase* pouvant prendre deux valeurs : ***'FREE'*** modèle en attente de message, ***'BUSY'*** modèle traduisant une activité du modèle ;
2. La variable d'état *sigma* $\in \mathbb{R}^{+}_{\infty}$ représentant la durée de vie d'un état donné du MA *"Net"*.

A l'arrivée de message, le modèle en état initial (***'FREE'***, INFINITY) évolue par la fonction δ_{ext} vers un état de transition (***'BUSY'***, 0). Dans cet état, un message, traité par l'algorithme de

routage, est généré grâce à la fonction de sortie λ vers le port de sortie Out 1. La fonction δ_{int} permet au modèle de retourner à son état initial.

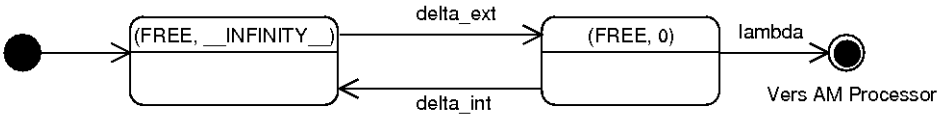


FIG. 4.7*: Graphe d'états du MA "*Net*"*

4.1.4.4 Le MA "*Flash*"

Ce modèle est défini dans un double objectif :

* stocker différentes valeurs propres au noeud ;

– intégrer de nouveaux types de message dans le cadre d'une non définition préalable.

Il possède deux variables d'états :

1. *phase* pouvant prendre deux valeurs : ***'FREE***', modèle en attente de message, ***'BUSY***', modèle traduisant une activité du modèle ;
2. La variable d'état *sigma* $\in \mathbb{R}_{\infty}^{+}$ représentant la durée de vie d'un état donné du MA "*Flash*".

Ce modèle n'est pas essentiel au fonctionnement du "*Processor*". Dans le cadre de notre approche et pour souci de "modifiabilité", il permet de faire évoluer le MA "*Processor*" sans modifier la définition basique du composant. Dans l'évolution du modèle et dans des perspectives de recherche, le MA"*Flash*" autorise la gestion du type de messages supplémentaires. Le développeur pourra utiliser la définition basique du MA "*Processor*", et rajouter la gestion de nouveaux types de messages. Plutôt que de redéfinir le MA "*Processor*", nous proposons le modèle "*Flash*" pour gérer les modifications, dans un contexte de personnalisation du modèle. En effet, à l'arrivée d'un type de message inconnu au niveau du MA "*Processor*", ce dernier est directement redirigé vers le MA

"Flash" qui va vérifier s'il ne contient pas ce type dans un registre défini par le developpeur. Dans notre application ce modèle existe mais ne sera pas utilisé, étant donné que nos messages sont typés par des valeurs connues par le système.

4.1.5 Le modèle énergétique

La consommation énergétique est un point essentiel dans la conception des capteurs. Ce modèle est relié à tous les modèles pouvant consommer de l'énergie dans le MC "*Sensor*", c'est-à-dire le MA"*COM*", le MA"*Processor*" et le MA"*Sensorboard*".

4.1.5.1 Présentation du modèle linéaire.

Pour la représentation de la consommation d'énergie, nous utilisons un modèle linéaire décrit par [Park et al., 2001, Savvides et al., 2001]. Dans le modèle linéaire, la batterie est considérée comme un stockage linéaire du courant. La capacité maximale de la batterie ne tient pas compte du taux de déchargement. Le modèle simple de batterie autorise l'utilisateur à visualiser l'efficacité de l'application en fonction de la consommation énergétique. La capacité restante C après une opération durant un temps t_d peut être exprimée par l'équation suivante :

$$C = C' - \int_{t=t_0}^{t_0+t_d} I(t)dt$$

où *C'* est la capacité précédente et $I(t)$ est le courant instantané i consommé par le circuit à un temps t. Le modèle linéaire assume que *I(t)* restera le même durant le temps t_d, si l'opération du noeud ne change pas durant la durée t_d.

Il existe d'autres modèles de batterie prenant en compte le taux de décharge et/ou prenant en compte le phénomène de ralaxation [Park et al., 2001].

4.1.5.2 Le MA "*Battery*"

Sur la Figure 4.8, nous présentons le MA *"Battery*". Il est composé de trois ports d'entrée et d'un port de sortie. Les quatres ports d'entrée vont servir à la réception des messages de consom-

mation énergétique de chaque composant au moment de chaque tâche réalisée par ces derniers. Le port 1 va recevoir les messages du MA *"COM*", le port 2 acceuille ceux venant du MC "*Process*", le port 3 acceuille la consommation du MA "*SensorBoard*". Il est important de préciser que les consommations électriques pour le MA "*Memory*" seront directement réalisées par le MC *"Process*". En effet, nous considérons que l'action d'écriture ou de recherche de données est faite par le processeur et que par conséquent cette consommation sera transmise par ce dernier.

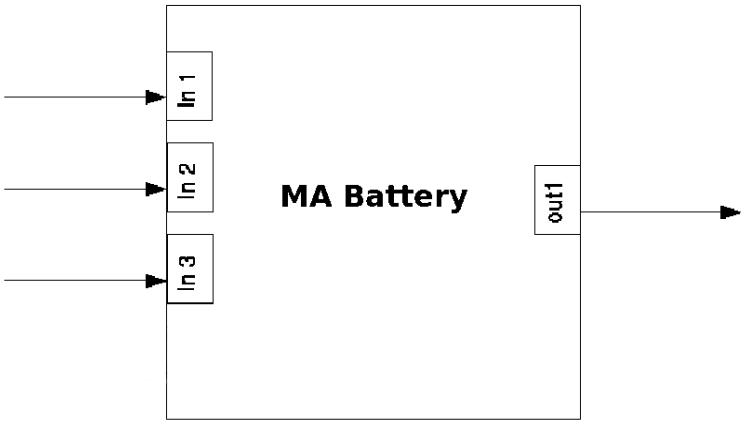


FIG. 4.8*: MA DEVS "*Battery*"*

La description dynamique du MA "*Battery*" est faite par le biais du graphe d'états du modèle énergétique illustré par la Figure 4.8.

Ce modèle a pour but de représenter la consommation d'énergie durant la simulation. Il possède deux variables d'états :

1. *phase* pouvant prendre trois valeurs : ***'FREE***' modèle en attente de message, ***'BUSY'*** modèle traduisant une activité du modèle, ***'DEAD***' modèle signifiant l'épuisement complet des capacités énergétiques ;
2. La variable d'état *sigma* $\in \mathbb{R}^{+}_{\infty}$ représentant la durée de vie d'un état donné du modèle atomiqe "*Battery*".

A l'arrivée du message, le modèle en état initial (***'FREE***', INFINITY) évolue par la fonction δ_{ext} vers un état de transition (***'BUSY***', 0). Dans cet état, le champ ***Conso*** du message contient la

valeur de consommation du composant qui a envoyé le message. Une valeur est décrémentée à l'intérieur du modèle : EnergyValue selon les caractéristiques du modèle linéaire (*cf* 4.1.5.1). La fonction λ ne sera utilisée que si l'attribut EnergyValue atteint 0. La fonction λ génère alors un message de type ***DEAD*** qui sera envoyé au MC "*Process*", qui par effet de diffusion va plonger le MA "*COM*" vers un état irréversible ***'DEAD'***, n'autorisant plus aucune communication.

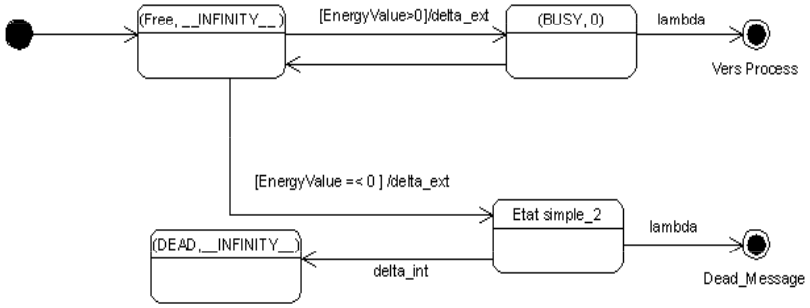


FIG. 4.9*: Graphe d'états du MA "*Battery*"*

Nous utilisons des données relatives à des consommations énergétiques en fonction du composant matériel utilisé dans la table de référence de [Crossbow-Technology, 2006, Hill et al., 2000a] que nous représentons sur la Figure 4.3.

SPECIFICATIONS DU SYSTEME		
Courants		**Exemple de rapport cyclique**
Processeur		
Courant (opération)	8 mA	1
Courant (passif)	8 µA	99
Radio		
Courant en réception	8 mA	0.75
Courant en transmission	12 mA	0.25
Courant en état passif	2 µA	99
Mémoire		
Ecriture	15 mA	0
Lecture	4 mA	0
Passif	2 µA	100
Sensor Board		
Courant (opération)	5 mA	1
Courant (passif)	5 µA	99
Calcul de l'énergie en mA-hr		
Processeur		0.0879
Radio		0.0920
Mémoire		0.0020
Sensor Board		0.0550
Courant total utilisé (mA-hr)		**0.2369**
Capacité de la batterie (mA-hr)		Durée de vie de batterie (mois)
250		1.45
1000		5.78
3000		17.35

TAB. 4.3*: Tableau de la consommation en fonction du composant*

Ce tableau nous précise les différentes consommations en fonction d'exemple de rapport cyclique (*Duty cyle*). Le rapport cyclique est la proportion de temps durant laquelle un composant est actif [Hohlt et al., 2004]. Supposons qu'un composant agisse durant 1 seconde et que pendant 99 secondes il soit éteint et qu'ensuite il refonctionnne pendant une seconde et ainsi de suite. Sur 100 secondes le Duty cycle sera égal à 1% selon le rapport entre :

$$D = \frac{\tau}{T}$$

où :

– *D est le* rapport cyclique *;*

– τ est le durée d'activité ;

– T est la période de fonctionnement du système (passif + actif)

Les concepteurs nous proposent donc des courants en fonction des temps d'utilisation. Nous utilisons ces valeurs pour fournir les données de consommation à notre modèle énergétique. Le temps

d'utilisation sera différent en fonction de l'activité du composant. Les valeurs seront décrémentées à la capacité énergétique de la batterie selon la formule de diminution de cette capacité énergétique (*cf* 4.1.5.1). Nous fournissons des temps constants d'utilisation, implémentés dans le MA "*Battery*" à nos composants en fonction des actions qu'ils ont à traiter, résumé dans le Tableau 4.4.

Composants	τ (s) pour $T = 4$ *(s)*	rapport cyclique
Processeur	0,04	1%
COM (receipt/trasnmit)	1/1,33	25%/75%
Sensor Board	0,04	2,5%
Memory	0,01	0,25%

TAB. 4.4*: Temps d'action en fonction de l'action à réaliser*

Une capacité de 3000 mA-hr est la capacité de deux piles de type AA. La durée de vie est estimée selon la capacité énergétique utilisée ; la durée de vie est importante si la capacité est importante.

4.1.6 Le modèle de mémoire

Dans notre modélisation nous séparons la mémoire propre au processeur (MA "*Flash*") et la mémoire permettant de stocker des informations provenant des senseurs environnementaux. Le but de ce modèle illustré par la Figure 4.10 est de pouvoir stocker des informations environnementales provenant des senseurs quand le capteur sans fil n'est pas en mode de communication. Le MA "*Memory*" est accessible par deux types de messages : *stockData* et *CollectData*. Le premier type de message provenant de la détection environnementale, le deuxième provient d'une communication extérieure ayant pour origine la station de base désirant collecter des informations contenues en mémoire.

Le MA "*Memory*" possède un port d'entrée In1 accueillant les messages provenant du MC "*Process*" et un port de sortie Out1 pour lui renvoyer des données.

FIG. 4.10*: MA DEVS "*Memory*"*

Ce modèle illustré par la Figure 4.11 a pour but de représenter la mémoire durant la simulation. Il possède deux variables d'états :

1. *phase* pouvant prendre deux valeurs : ***'FREE***' modèle en attente de message, ***'BUSY'*** modèle traduisant une activité du modèle ;
2. La variable d'état *sigma* $\in \mathbb{R}_{\infty}^{+}$ représentant la durée de vie d'un état donné du modèle atomiqe "*Memory*".

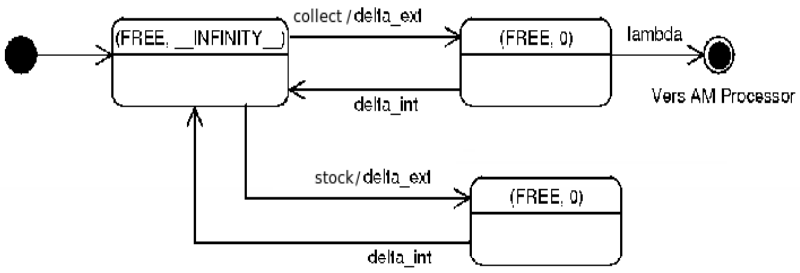


FIG. 4.11*: Graphe d'état du MA "*Memory*"*

Ce modèle est implémenté par défaut mais ne sera pas utilisé dans notre application. Nous avons défini ce modèle par souci de "réutilisabilité" dans de futures applications.

4.1.7 Le modèle de détection environnementale

Cette partie présente notre approche originale dans la prise en compte des interactions entre carte "capteurs" et l'environnement physique. Les capteurs ont pour nature de détecter des valeurs de changements environementaux mais également de communiquer. Cette action est réalisée par la carte "capteurs". Dans notre approche, nous distinguons deux types de modèles : le modèle représentant l'ensemble de la carte "capteurs" représenté par le MA "*Sensorboard*" et ensuite un modèle contenant les données environnementales représenté par le MA "*Env*". Nous n'avons pas voulu définir ces deux comme un modèle couplé pour deux raisons :

- dans un souci de représentativité, le modèle atomique "*Env*" est extérieur au MC "*Sensor*" car, selon nous, celui-ci ne peut pas être considéré comme un composant matériel ;
- l'ajout du MA"*Env*" offre la possibilité de greffer sur chaque modèle, un environnement physique propre pour la génération des données environnementales, locales par exemple. Nous considérons naturellement qu'un MA "*Env*" général peut être appliqué à tous les modèles, mais nous offrons la possibilité de pouvoir avoir un MA "*Env*" pour chaque capteur.

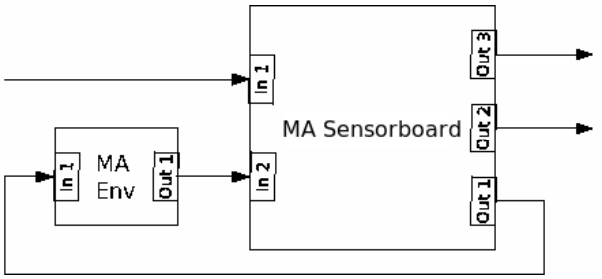


FIG. 4.12*: MA DEVS "*SensorBoard*"*

Nous présentons le diagramme d'état du MA "*SensorBoard*". Ce modèle illustré par la Figure 4.13 a pour but de représenter les interactions entre le capteur et son environnement. Il possède deux variables d'états :

1. *phase* pouvant prendre cinq valeurs :
   - ***'FREE***' modèle en attente de message ;

- ***'BUSY'*** modèle traduisant une activité du modèle ;
- ***'WAIT'*** lors de la phase de collecte d'informations environnementales ;
- ***'GO'*** signifiant l'envoi de données vers le MC "*Process*" si une valeur fixée ne dépasse pas un certain seuil ;
- ***'DEAD'*** signifiant l'envoi d'un message de type *DEAD* vers le MC *"Process*" si une valeur a dépassé un certain seuil.

2. La variable d'état *sigma* $\in \mathbb{R}^{+}_{\infty}$ représentant la durée de vie d'un état donné du MA "*Sensor-Board*".

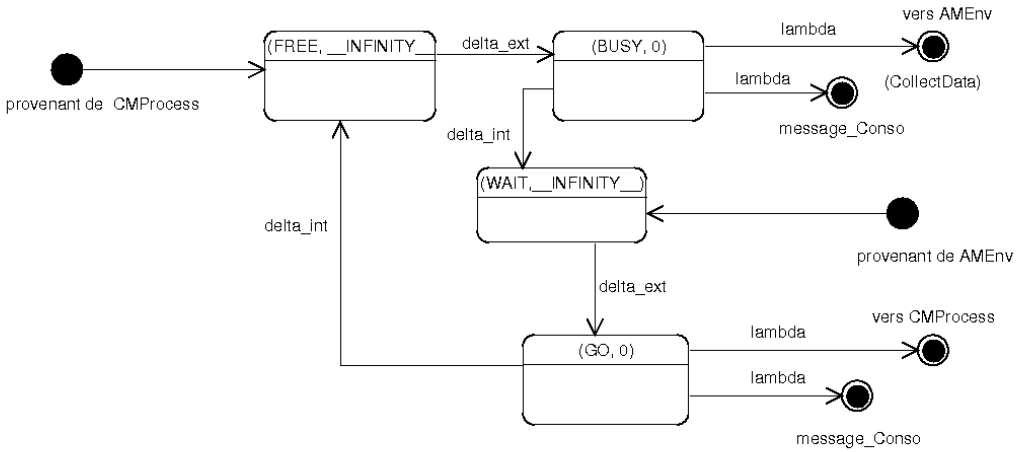


FIG. 4.13*: Graphe d'états du MA "*SensorBoard*"*

Lors de la phase de réception d'un message, l'état du modèle ***'FREE'*** évolue vers un état ***'BUSY'***, signifiant l'envoi d'un message vers le MA "*Env*" pour la collecte de données. La transition interne fait évoluer le modèle vers un état ***'WAIT'***, autorisant exclusivement la réception de messages environnementaux du MA *"Env*". A la réception d'un message environnemental, de température, la fonction de transition externe autorise le passage à l'état ***'GO'*** si la valeur est inférieure au seuil fixé dans le MA "*Sensorboard*", déclenchant la sortie du message de données vers le MC *"Process*" et du message de consommation vers MA "*Battery*". La transition interne fait

évoluer le modèle vers l'état initial ***'FREE***'. Dans le cas où la valeur collectée dépasse le seuil fixé, l'état évolue vers un état irréversible, envoyant un message de type *DEAD* au MC"*Process*" ; ce dernier transmettant le message de type *DEAD* au MA "*COM*" qui évolue dans une phase d'interruption des communications.

Ce modèle a pour but de générer des comportements environnementaux. Nous entendons par comportement environnemental, la possibilité pour chaque capteur d'aller chercher des données sur son environnement directement dans le MA "*Env*". La notion de seuil abordée est une particularité de notre approche. En effet dans la problématique des feux de forêt et de leurs effets destructeurs, nous voulons que si une valeur de température est trop importante, le modèle évolue vers un stade critique ***'DEAD'*** représentant un capteur brûlé et détruit par l'incendie.

Nous venons de représenter les composants matériels d'un capteur. Il est maintenant essentiel de définir les processus de communication. Nous allons détailler les processus de communications intra et inter-nodales de notre approche.

4.2 Communication entre les noeuds

Le processus de communication définit les régles par lesquelles le transfert de données s'effectue entre les noeuds. Il est important de préciser à ce niveau la méthode de communication entre les noeuds. Au niveau le plus haut d'abstraction, c'est à dire au niveau réseau, les entités vont devoir s'échanger des données pour transférer la communication vers la station de base. Pour ce haut niveau de description, nous proposons, comme base de représentation et pour une meilleure compréhension de notre approche, un réseau à trois noeuds comme illustré sur la Figure 4.14.

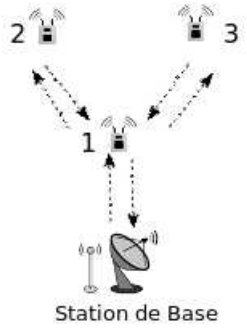


FIG. 4.14*: Réseau à trois noeuds*

Sur cette figure nous représentons trois noeuds et une station de base. Les communications sont également représentées. Dans notre exemple, les noeuds 2 et 3 ne peuvent pas communiquer entre eux car nous considérons dans ce cas précis qu'ils sont trop éloignés.

4.2.1 Approche des communications inter-noeud

Nous allons détailler le processus de base lors de la communication entre les noeuds dépendant de la couche MAC (*cf* 2.1.2). En effet, même si nous focalisons nos travaux sur les protocoles de routage, notre approche aurait été incomplète sans la prise en compte du protocole MAC. La Figure 4.15 va représenter le mode S-MAC [Ye et al., 2002b], protocole MAC dédié aux réseaux de capteurs sans fil. Ce protocole MAC autorise non seulement une interaction fiable entre les noeuds mais aussi par le biais d'un échange de messages balise, une économie d'énergie dans les entités du réseau encadrant les phases d'activité et de passivité. Cette communication va se passer en deux phases :

1. phase d'initialisation ou synchronisation ;
2. phase de fin de communication.

Nous proposons de représenter les communications entre ces quatres entités par un diagramme de séquence comme décrit sur la Figure 4.15.

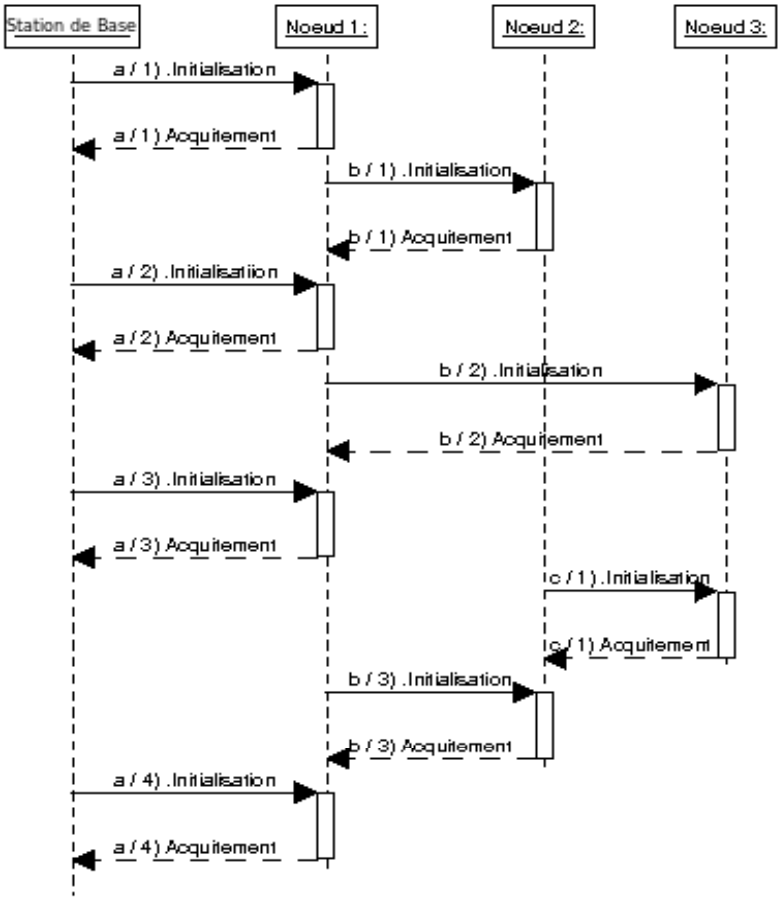


FIG. 4.15*: Diagramme de séquence général*

Dans notre exemple illustré par la Figure 4.14, les noeuds 2 et 3 ne peuvent communiquer directement avec la Station de Base. Pour cela, les deux noeuds trop distants vont utiliser le noeud 1 comme relais de communication. Nous voyons sur la Figure 4.15, que la Station de Base va initier la première communication par l'envoi d'un message d'initialisation et ensuite va attendre un message d'acquittement qui est identifié par le type *ACK* pour la fin de la transmission. Chaque communication entre noeud sera initialisée par le biais d'un message et se conclura également par l'envoi d'un message de fin de communication. Les communications entre les noeuds vont dépendre naturellement de la structure du réseau.

Nous proposons de préciser la communication plus particulière entre deux noeuds en utilisant un niveau plus bas de description, celui des composants.

4.2.2 Approche des communications intra-noeud

Nous avons précédemment décrit le processus de balises utilisé par le protocole S-MAC. Nous proposons d'affiner cette vue avec le cas d'un échange de données environnementales, permettant d'illustrer l'action de chaque composant. Pour cet exemple, le processus se décompose en réalité en trois phases.

1. phase d'initialisation ou synchronisation ;
2. phase d'échange d'informations ;
3. phase de fin de communication.

Les différentes phases que nous allons décrire plus précisemement, représentent la méthode utilisée par le réseau pour communiquer (*cf* 4.2.1). Sur la Figure 4.16, nous représentons l'échange de messages entre le Noeud 1 et le Noeud 2.

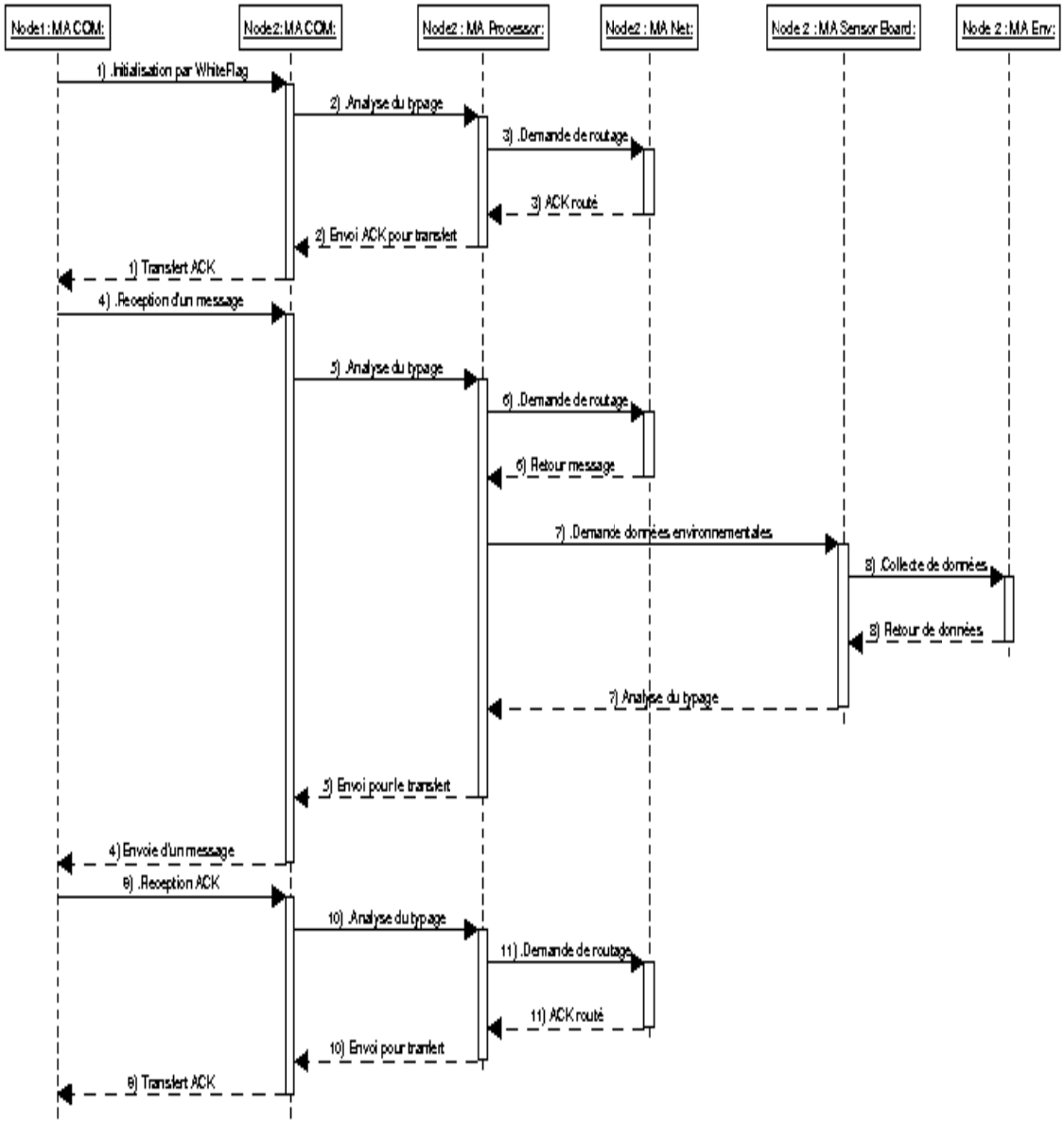


FIG. 4.16*: Diagramme de séquences détaillé*

4.2.2.1 Phase d'initialisation

L'initialisation d'un transfert de message se fait par le biais d'un message de type *WhiteFlag* qui est l'analogie du message HELLO bien connu dans le domaine du réseau. Ce type de message

est utlisé par tous les noeuds pour initier une communication. Le message *WhiteFla*g est généré par le Noeud 1 qui souhaite débuter une conversation avec le Noeud 2. Ce message est reçu par le Noeud 2. Le MA "*COM*" va bloquer son état n'autorisant plus aucune communication durant cette phase de traitement de l'information. Ce message est transmis au modèle MA "*Processor*" qui va analyser le typage. Le MA "*Processor*" va diriger le MA "*Net*" qui va fournir la destination par le biais du protocole de routage. Cette action réalisée, le type de message est renvoyé, avec comme destination le Noeud 1 et typé *ACK*. Ce type *ACK* certifie que le message a été traité et routé comme nous l'avons défini précédemment (*cf* 4.2.1) . Le message de type *ACK* est retourné vers le MA "*Processor*" qui, après analyse du typage, va être redirigé vers le MA "*COM*" . Le message arrive sur le MA "*COM*" qui analyse la destination et qui le renvoie dans notre exemple au Noeud 1. A partir de cette initialisation, la transmission des données va pouvoir commencer.

4.2.2.2 Phase d'échange d'informations

Le noeud 1 va recevoir le message de type *ACK* provenant du Noeud 2 certifiant que ce dernier est prêt pour la communication. Le Noeud 1 va à la réception modifier le typage et générer un message de type *Collect* et va être renvoyé vers le Noeud 2. Le Noeud 2 va recevoir ce message et recommencer le même processus d'analyse que dans la phase d'initialisation, avec la nécessité en premier lieu de donner une destination par le biais du MA "*Net*". La différence est que le MA "*Net*" ne va pas modifier le typage et retourner ce message vers le MA "*Processor*" après avoir fourni les informations de routage. Le MA"*Processor*" va recevoir ce message provenant de MA "*Net*" avec un type *Collec*t. Le MA "*Processor*" va orienter le message vers le MA "*SensorBoard*" pour une collecte de données environnementales.

4.2.2.3 Phase de fin de communication

Cette phase de communication utilise le MA *"Sensorboard*" et le MA *"Env*". En effet , pour permettre le transfert d'informations environnementales, notre modèle de capteur va puiser les informations dans le MA *"Env*". A l'arrivée du message, le typage sera analysé et le MA "*Sensor-Board*" permettra la collecte d'informations. Les informations collectées, une analyse des données

est faite pour vérifer le possible franchissement de seuil d'une données et dans tous les cas, le message sera redirigé vers le MA "*Process*or". La fin des communications sera signifiée par l'échange d'un dernier message de type *ACK*.

Nous venons de définir comment les communications sont réalisées dans le réseau. Les règles de communication du protocole S-MAC par le biais de balises (types *Whiteflag* et *ACK*) autorisent des phases d'activité et des phases de sommeil. Nous avons vu dans cet exemple que pour router les messages, un noeud utilise le MA "*Net*". Ce dernier a pour but, par le biais d'un algorithme de sélection, de déterminer la destination du message. Nous proposons dans la partie suivante de nous intéresser à cette notion d'algorithme de routage.

4.3 Algorithmes des protocoles de routage étudiés

4.3.1 Présentation

Dans cette partie nous allons aborder les notions essentielles pour comprendre le fonctionnement du système dans la phase de routage. La nature même du système se focalise sur la notion de réseau. La particularité des réseaux de capteurs sans fil repose sur la notion d'architecture distribuée, impliquant collaboration des entités pour accomplir la tâche du réseau (surveillance environnementale d'une zone). La mise en place et le maintien de l'architecture va reposer sur deux phases :

- la découverte du voisinage (les entités à portée radio).

- le maintien du voisinage.

La découverte du voisinage va se baser sur la notion de diffusion d'un message HELLO ou *WhiteFlag* initié par la station de base, qui, comme nous l'avons vu précédemment, permet également l'organisation des communications (protocole MAC).

Pour permettre des communications multi-sauts, les entités du réseau doivent connaitre leur voisinage. Cette phase de découverte est initiée par la station de base. La station de base est considérée comme le générateur source du message *WhiteFlag*. Elle va l'émettre vers les noeuds à portée radio. Le message arrivant sur le premier noeud sera ensuite diffusé à toutes les entités

voisines. Par effet de diffusion (*cf* 2.2), ce message sera émis de proche en proche dans tout le réseau. Ces messages vont permettre non seulement la mise en place des règles d'accès et de partage du media (*cf* 4.2.1) mais également l'établissement des différentes listes de voisins nécessaires dans notre cas dans le routage. Le maintien du voisinage, c'est à dire la mise à jour régulière du voisinage, est un élément essentiel pour ces structures. Ce dernier consistera donc à un classement de ce voisinage selon l'algorithme de routage. Ce classement est contenu dans la table de routage. En fonction des caractéristiques des voisins, chaque noeud estime et classe les meilleurs noeuds dans un ordre décroissant.

Nous devons également distinguer, dans le cadre de notre approche, la table de voisinage et la table de routage FT . Lors des communications il existe des messages de diffusion et de communications. Les messages de diffusion seront envoyés par la station de base pour la mise en place de la structure du réseau, que l'on nommera phase ascendante, *"one-to-many*", un vers tous. Aucun noeud ne doit être exclu et pour cela les noeuds doivent connaître leur voisinage. Tous les noeuds sont donc inclus dans la table de voisinage. Nous définissons comme table de routage FT, la classification des voisins issus de la table de voisinage selon des critères précis de sélection. La table de routage FT est utilisée majoritairement dans la phase descendante des messages, "*many-to-one*" pour l'envoi des données vers la station de base.

Nous rappelons que dans notre cas, nous n'étudierons que des protocoles dits "à plats" pour les raisons que nous avons énoncées précédemment *(cf 2.2.4)*.

Nous introduisons dans cette partie les protocoles que nous soumettrons à la simulation. Nous ferons d'abord une présentation de la technique de la diffusion directe qui est la référence dans le cadre de la famille des protocoles dits "plats", ensuite nous introduirons la définition des algorithmes de routage GBR, Xmesh. Nous proposons ensuite un nouvel algorithme de routage VOX.

4.3.2 La diffusion directe

Tous les protocoles que nous étudierons dérivent du concept de diffusion directe présenté par [Intanagonwiwat et al., 2000]. La diffusion directe est l'un des systèmes de routage les plus populaires dans le domaine des réseaux de capteurs sans fil. Il appartient à la famille des protocoles

dits "plats" et dont beaucoup de protocoles sont des variantes plus élaborées. Dans la diffusion directe, on distingue trois phases dans le routage :

1. propagation d'un message ;
2. installation du gradient ;
3. livraison des données.

Cette notion de diffusion directe est expliquée sur la Figure 4.17 **a)**, **b)**, **c)**. La phase propagation d'un message va être l'envoi d'un message à travers le réseau. Ce message va permettre l'établissement de la structure du réseau. La station de base génère périodiquement un message et l'envoi à un premier noeud. Ce noeud reçoit ce message et le diffuse à son voisinage direct, établissement des tables de voisinage. Cette propagation sera continue jusqu'à l'établissement et l'installation d'un gradient et la mise en place de la table d'envoi . Dans le cas de la diffusion directe, le gradient est estimé en fonction du taux de données reçues par chaque noeud et de la fraicheur des messages reçus. Ainsi chaque noeud maintient une table contenant tous ses voisins, classés de manière à ce que le premier noeud de cette table soit le plus fiable pour une transmission de données, c'est à dire avec le taux de données le plus élevé et un rafraîchissement fréquent. La dernière phase correspond à l'envoi de données en fonction du gradient établi pour chaque noeud. Le noeud source, celui proche de l'événement envoie ses données au noeud le plus fiable. Ce dernier fait de même jusqu'à que les données arrivent jusqu'à la station de base.

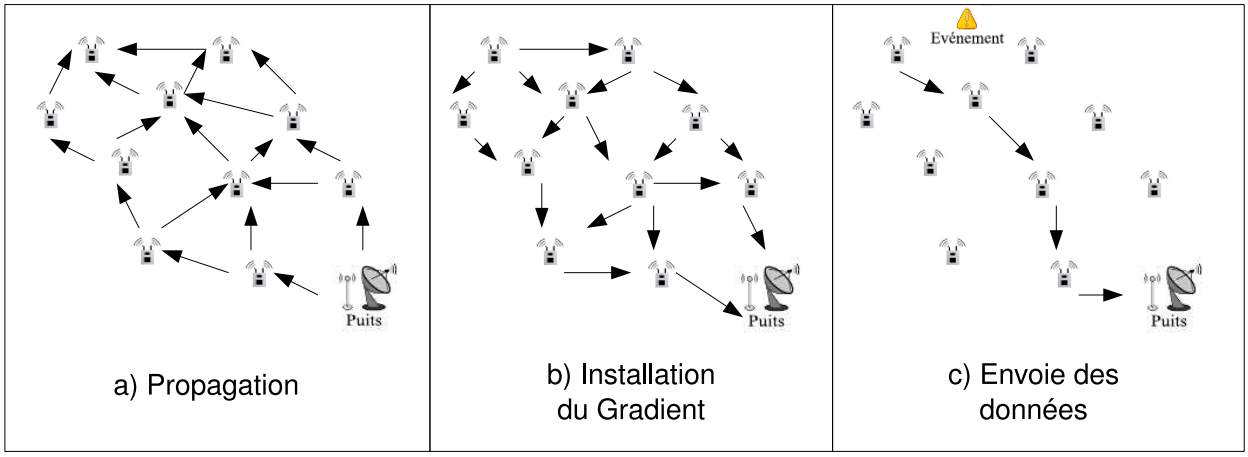


FIG. 4.17*: Schéma général de la diffusion directe*

Ce système de routage permet à chaque noeud de déterminer quel sera le voisin le plus fiable pour la transmission des données vers la station de base. Il existe des variantes à la diffusion directe, plus pertinente, basé sur une classification différente du voisinage. L'élaboration de ces tables de routage se base sur des métriques différentes. Nous utilisons le terme métrique pour une valeur qui permet de définir, entre deux noeuds concurrents, le noeud qui aura la place de voisin idéal dans la table de routage. Ce dernier sera choisi pour router les informations.

4.3.3 Gradient basé sur le nombre de sauts

La métrique, utilisant le nombre de sauts effectués depuis la station de base, est la base du protocole GBR (Gradient-Based Routing) proposé par [Schurgers et Srivastava, 2001]. Ce protocole est une variante de la diffusion directe. L'idée du protocole GBR est la mémorisation du nombre de sauts utilisés depuis la station pour atteindre un noeud. Ainsi chaque noeud peut calculer un paramètre appelé hauteur du noeud, dans notre approche ***Sauts***, qui est le minimum de sauts que doit utiliser un noeud pour atteindre la station de base. Pour cela, la station de base se charge de mettre en place des chemins qui relient les capteurs jusqu' à elle-même. Pour construire les différents chemins, la station de base diffuse sur le réseau un message qui déclare une métrique avec un coût à zéro. Les noeuds les plus proches qui reçoivent le message mettent à jour la métrique, puis le retransmettent et ainsi de suite. Quand un capteur veut émettre une donnée, il envoie alors le message au noeud qui a déclaré la métrique la plus faible pour aller à la station de base, et le message est alors retransmis de noeud en noeud toujours en suivant la route qui possède alors la plus faible métrique comme illustré sur la Figure 4.18.

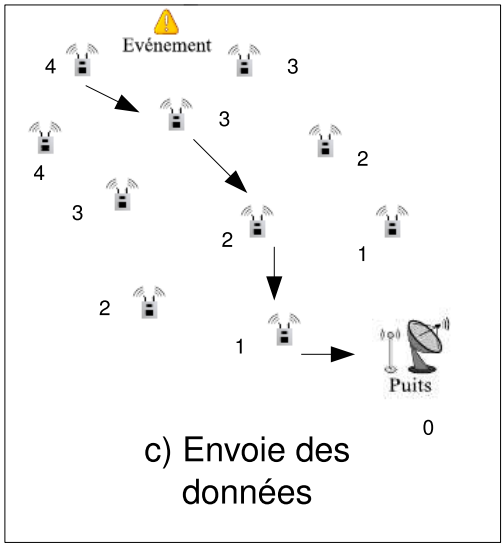


FIG. 4.18*: Routage basé sur le gradient*

Les auteurs précisent qu'à niveau équivalent, c'est à dire si deux noeuds possédent le même nombre de sauts, la sélection du voisin, dans le cas de l'envoie de données, se fait de manière aléatoire. Le routage va dépendre du processus de sélection du voisin idéal, traduit dans l'algorithme 1, se basant sur une seule métrique comme critère de sélection avec :

- N_i un noeud qui est soumis à concurrence pour la place de noeud idéal dans la table d'envoie FT du noeud N_k recepteur *;*
- Hop_i est le nombre de sauts pour atteindre la station de base.

Algorithme 1 Algorithme de sélection du voisin selon GBR

Algorithme selectNeighbor_*GBR*(N_i)
Début
Pour *x allant de 0 à* $FT[max]$
Si $N_i \notin FT[x]$ alors
éviction de la table(N_i)
Sinon $(Hop_{Ni} < Hop_{FT[max]})$ alors
éviction de la table (N_i)
Fin Si
Si $(Hop_{Ni} > Hop_{FT[x]})$ ***//métrique 1***
$cle \leftarrow FT[x]$
insertion dans la table(N_i)
mise à jour de la table(cle, FT)
Fin Si
Si $(Hop_{Ni} = Hop_{FT[x]})$ *alors*
alors
$cle \leftarrow N_i$
$cle2 \leftarrow FT[x]$
$voisin \leftarrow (slectionalatoireentrelesnoeuds(cle, cle2))$
insertion dans la table(voisin)
mise à jour de la table $(noeud\,restant, FT)$
Fin Si
Fin Pour
insertion dans la table(N_i, $FT[max+1]$)
Fin

Le but de l'algorithme de routage selon GBR est de gérer la table de routage à chaque entrée d'un nouveau message provenant d'un noeud voisin. L'algorithme va parcourir la table de routage et vérifier si le noeud entrant a un nombre de sauts inférieur aux éléments de la table. Si ce cas se produit, la table de routage est mise à jour par l'insertion du nouveau noeud. Par contre si le noeud entrant possède le même nombre de sauts qu'un noeud déjà présent dans la table, une fonction détermine aléatoirement quel noeud va prendre cette place dans la table.

Nous voyons que cet algorithme est assez simple. Nous allons voir à présent un algorithme de routage plus élaboré basant la sélection de ses voisins selon une double métrique.

4.3.4 Le protocole *Reliable route protocol* ou Xmesh

Le protocole Reliable route protocol présenté par [Woo et al., 2003a], plus connu sous le nom de Xmesh depuis son implémentation sur les plates-formes MICA2 [Crossbow-Technology, 2006],

est également une technique de routage basée sur le gradient. Cependant il met en oeuvre plusieurs principes permettant un routage efficace semblable en plusieurs points au protocole Ad-hoc On Demand Vector [Farooq et al., 2004]. Premièrement, la métrique utilisant le nombre de sauts est également présente dans ce protocole. Deuxièmement les auteurs mettent en avant une procédure de gestion du voisinage plus élaborée que le protocole *GBR*. En effet, le processus de gestion des messages entrants permet non seulement d'éviter les redondances dans les messages mais assure également une qualité de transmission en permettant une meilleure classification des messages, en ajoutant un nouveau critère de sélection qui est la qualité des liens. La qualité des liens représente un pourcentage entre les messages envoyés et les messages reçus. Nous expliquons le processus par la Figure 4.19.

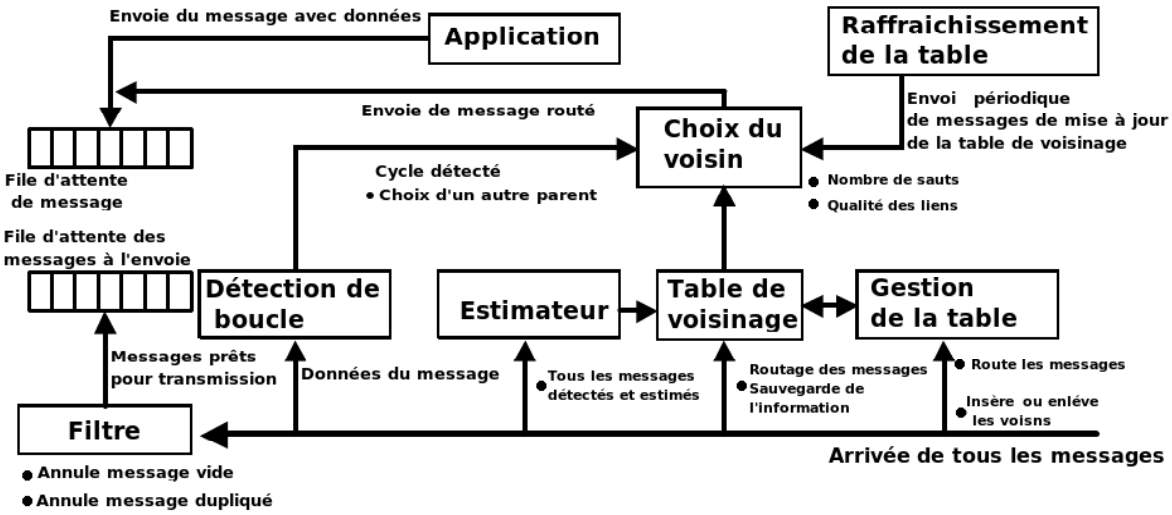


FIG. 4.19*: Diagramme illustrant les différents composants du protocole* Xmesh

Ce protocole se compose de différentes phases que nous détaillons :

– la première étape consiste à permettre la mise à jour de la table de routage en fonction des messages entrants. Cette sélection permet de choisir parmi les voisins lequel sera le premier dans la liste de routage. A ce niveau il existe un processus de confirmation ou d'éviction de voisin dans la table de routage. Le nombre de messages provenant d'un noeud va être également comptabilisé. Si durant un certain temps, un noeud n'émet plus de messages, il chute dans la hiérarchie des voisins jusqu'à être éliminé. A contrario, si un noeud vient à

produire beaucoup de messages, celui-ci est confirmé dans la hiérarchie ;

- la seconde étape consiste à fournir aux messages entrants les informations de routage avec une sauvegarde des données nécessaires ; les messages sont ensuite envoyés dans la file d'envoi des messages ;
- la troisième étape consiste à déterminer les cycles. Cette détection va se faire en déterminant l'identité de l'émetteur. En effet, en fonction des informations transmises, le noeud récepteur peut détecter une boucle. Une boucle peut se produire lorsqu'un message de type ACK est perdu et que le noeud émetteur n'a pas reçu l'aquittement du noeud récepteur. Le noeud émetteur va continuer à émettre jusqu'à recevoir un acquittement. Le noeud récepteur qui a déjà reçu le message, mais dont le message d'acquitement n'a pas été reçu , détecte une boucle et dans ce cas déclenche une nouvelle sélection de parents pour l'envoie des données. Si ces messages sont des messages dupliqués ou bien des messages vides, c'est à dire sans données, ils sont evincés ;
- la technique de rafraîchissement de la table permet d'envoyer périodiquement des messages de mise à jour, permettant de fiabiliser les tables de routage de chacun des voisins ;
- une fois tous ces processus réalisés, les messages sains sont envoyés ainsi que les données des applications ;
- les messages sont stockés dans une file d'attente.

Il est important de remarquer que ce protocole contient deux files d'attente. Les auteurs soulignent que les messages dit d'infrastructure sont plus nombreux que les messages d'applications. Pour cela la file d'attente contenant les messages d'applications est prioritaire et prévaudra sur la hiérarchie de transmission.

A présent, nous abordons la notion de métrique. Nous savons que la métrique utilisée en priorité est le nombre de sauts. La différence avec le protocole GBR se fait sur le choix du parent à niveau de sauts équivalents. En effet, dans le protocole Xmesh, le choix va se porter sur le voisin ayant la meilleure qualité de lien et la plus forte fréquence de demande d'insertion dans la table. La qualité de lien représentée par un pourcentage, va servir de seuil et de moyen d'éviction des voisins. Un voisin idéal possède une qualité de lien supérieure à 75% sachant que sa fréquence

est supérieure. Les noeuds dont la qualité de lien est inférieure à 40% sont évincés de la table de routage. Cette métrique permet donc de disposer en tête de la table de routage des noeuds assurant la meilleure transmision des données. Cette technique est traduite par l'algorithme 2 basé sur deux critères de sélection avec :

- N_i un noeud qui est soumis à concurrence pour la place de noeud idéal dans la table de voisinage FT du noeud N_k ;
- Hop_i est le nombre de sauts pour atteindre la station de base ;
- $Freq_{Ni}$ est la fréquence de demande sélection ;
- $LinkN_i$ représente la qualité des liens du noeuds vis à vis du noeud N_k ;

Algorithme 2 Algorithme de sélection du voisin selon Xmesh

Algorithme selectNeighbor_*Xmesh* **(N_i)**
Début
Pour *x allant de 0 à* $FT[max]$ par incrément de 1 faire
Si $N_i \notin FT[x]$ et que $(Link_{Ni} < 50)$ *ou* $(Hop_{Ni} < Hop_{FT[max]})$ alors
eviction de la table(N_i)
Sinon
$Freq_{Ni} \leftarrow Freq_{Ni} + 1$
Si $(Hop_{Ni} > Hop_{FT[x]})$ alors **//métrique 1**
$cle \leftarrow FT[x]$
insertion dans la table(N_i)
mise à jour de la table(cle, FT)
Fin Si
Si $(Hop_{Ni} = Hop_{FT[x]})$ *alors*
Si $(Freq_{Ni} > Freq_{FT[x]})$ alors
Si $(Link_{Ni} > 75)$ et $(Link_{Ni} > (Link_{FT[x]})$ alors **//métrique 2**
$cle \leftarrow FT[x]$
insertion dans la table(N_i)
mise à jour de la table(cle, FT)
Fin Si
Fin Si
Fin
Fin Si
Fin Pour
éviction de la table(N_i)
Fin

Nous observons donc l'ajout d'une seconde métrique dans la sélection du voisin. Naturellement la fréquence des messages d'un noeud voisin est également prise en compte, ce qui est déjà une différence majeure avec le protocole GBR, mais la pertinence du protocole se révèle surtout dans la considération de la qualité de transmission, exprimée par le lien, qui unit deux capteurs dans leur transmission. De plus à la différence de GBR, si un noeud ne posséde pas les conditions requises, il sera evincé de la table de routage.

Après l'étude de ces deux protocoles de routage, il apparait de réelles capacités. Xmesh est plus évolué que GBR, cependant Xmesh ne tient pas compte de la consommation énergétique, qui est un paramètre essentiel dans les réseaux de capteurs. Pour cela nous proposons la définition d'un nouvel algorithme de routage basé sur le protocole Xmesh.

4.3.5 Proposition du protocole VOX

L'inquiétude récurrente dans les réseaux de capteurs sans fil est la conservation de l'énergie. Certains protocoles comme *EAR* [Shah et Rabaey, 2002], proposent un routage basé sur la consommation énergétique. Seulement un noeud ayant un bon niveau énergétique ne garantit pas une transmission correcte des données. La transmission correcte des données peut être validée par une qualité des liens comme le propose le protocole *Xmesh*. Cependant ce dernier n'assure pas une bonne répartition énergétique. Partant de ce constat, pour transmettre correctement des informations sans pour autant compromettre l'équilibre énergétique du réseau, nous proposons une variante du protocole *Xmesh* prenant en compte ce paramètre. La technique de gestion des tables de routage et le processus pour évincer des redondances proposés par le protocole Xmesh sont pertinents. De plus, le protocole garantit la qualité de la transmission en favorisant les noeuds ayant la meilleure qualité de liens, seulement il ne prend pas en compte la sauvegarde de l'énergie. Nous voulons favoriser la conservation de l'énergie. Pour conserver de l'énergie, nous proposons de ne pas utiliser en priorité le noeud avec le lien le plus fort mais de nuancer cet aspect, en favorisant la concurrence entre les noeuds. Un noeud ayant une qualité de lien de 90% et un noeud avec une qualité de lien de 89% peuvent être considérés comme équivalents pour assurer une transmission qualitative des données. Pour cela nous choisissons de fixer une règle supplémentaire au protocole

Xmesh :

- ***condition 1 : deux noeuds ayant une différence de qualité de liens inférieure à 5% sont portés à concurrence ;***
- ***condition 2 : en respectant la condition précédente, le noeud ayant la plus forte capacité énergétique sera favorisé pour le routage jusqu'à la prochaine mise à jour.***

Nous proposons, à partir des conditions précédemment cités, un nouvel algorithme de sélection du voisinage, Variant Of Xmesh (VOX) favorisant la prise en compte de ces nouveaux critères en sachant que :

- N_i un noeud qui est soumis à concurrence pour la place de noeud idéal dans la table de voisinage FT du noeud N_k ;
- Hop_i est le nombre de sauts pour atteindre la station de base ;
- $Freq_{Ni}$ est la fréquence de demande sélection ;
- $LinkN_i$ représente la qualité des liens du noeuds vis à vis du noeud N_k ;
- CN_i représente les capacités énégétiques ou énergie résiduelle du noeud.

Nous proposons pour le choix du voisin idéal l'algorithme suivant 3 :

Algorithme 3 Algorithme de sélection du voisin selon VOX

Algorithme selectNeighbor_VOX(N_i)
Début
Pour *x allant de 0 à* $FT[max]$ par incrément de 1 faire
Si $N_i \notin FT[x]$ et que $(Link_{Ni} < 50)$ *ou* $(Hop_{Ni} < Hop_{FT[max]})$
eviction de la table(N_i)
Sinon
$Freq_{Ni} \leftarrow Freq_{Ni} + 1$
Si $(Hop_{Ni} > Hop_{FT[x]})$ alors ***//métrique 1***
$cle = FT[x]$
insertion dans la table(N_i)
mise à jour de la table(cle, FT)
Sinon $(Hop_{Ni} == Hop_{FT[x]})$
Si $(Freq_{Ni} > Freq_{FT[x]})$ alors
Si $(Link_{Ni} > 75)$ alors ***//métrique 2'***
Si $(0 < Link_{FT[x]}$ - $Link_{Ni} < 5)$ alors
Si $(C_{Ni} > C_{FT[x]})$ alors ***//métrique 3***
$cle \leftarrow FT[x]$
insertion dans la table(N_i)
mise à jour de la table(cle, FT)
Fin Si
Sinon $(Link_{FT[x]}$ - $Link_{Ni} < 0)$ alors ***//métrique 2***
$cle \leftarrow FT[x]$
insertion dans la table(N_i)
mise à jour de la table(cle, FT)
Fin Si
Fin Si
Fin Si
Fin Si
Fin Si
Fin Pour
eviction de la table(N_i)
Fin

Nous voyons l'insertion de la métrique 2' avec insertion de la ***condition 1***, puis un test selon la ***condition 2*** qui correspond à la métrique 3 fonction de la capacité énergétique du noeud. La métrique 2 apparaît en fin d'algorithme.

Nous illustrons également les quelques modifications apportées au protocole Xmesh sur la Figure 4.20.

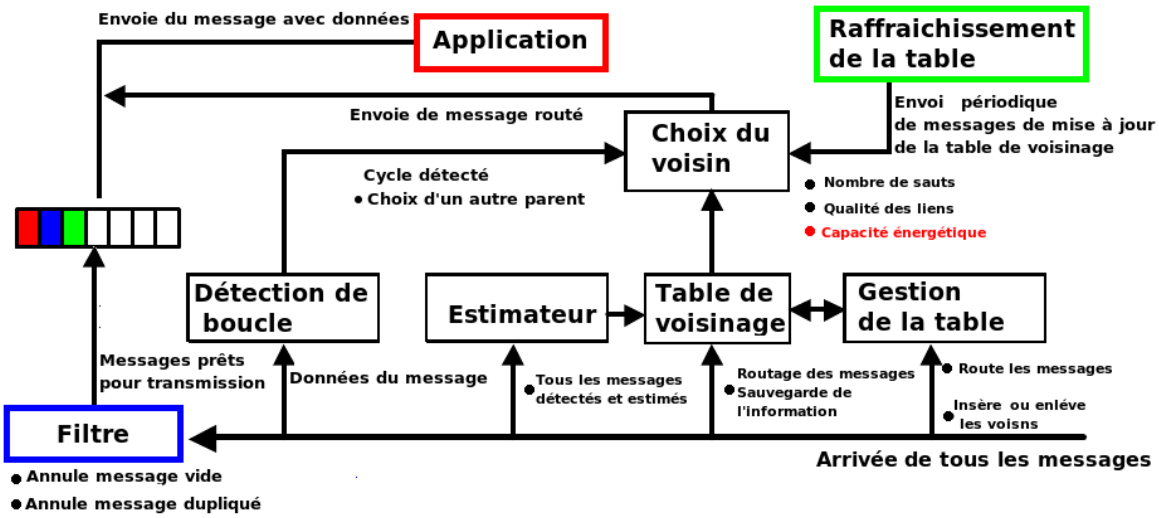


FIG. 4.20*: Structure du protocole Variant Of Xmesh*

La modification principale porte sur le processus de sélection des noeuds voisins. Cependant nous ajoutons une légère modification sur la gestion des messages. Nous ne possédons plus qu'une seule file d'attente. Dans notre approche, la classification des messages dans la file d'attente s'opère en utilisant la nature même du message ce qui donne l'ordre de prorité suivant :

$$Msg Applications > Msg de communication courante > Msg de mise\ jour$$

Cet ordre nous permet d'assurer une priorité dans la transmission des messages issus des capteurs, essentiels dans la surveillance environnementale. Nous voulons ainsi éviter les conflits et les possibles décalages lors de l'envoi de messages.

4.4 Conclusion

Nous venons de décrire une modélisation de capteurs par le biais du formalisme à événements discrets DEVS. Cette approche se base sur la définition des composants d'un capteur sans fil. Nous y proposons une approche de modélisation originale par la définition de tous les composants matériels et par la caractérisation d'un modèle environnemental pour la génération de scénario feu de forêt. Ces modèles sont intégrés au sein de notre application modulaire **DEVS-WSN**. Nous

nous sommes attachés à définir une structure indépendante du matériel offrant un aspect générique. De plus, nous fournissons des éléments pour autoriser l'évolution de notre application, notamment avec le MA "*Flash*" dans un souci de personnalisation de l'application. Nous avons également ajouté à ces travaux, l'introduction d'un nouvel algorithme de routage par le biais de l'algorithme VOX, variante du protocole Xmesh. Ce protocole se base sur trois paramètres : le nombre de sauts, la qualité des liens et la ressource énergétique. Le protocole VOX tend à optimiser des protocoles issus de la famille des protocoles dits "plats", que nous avons identifiés comme les plus efficaces dans le cadre de la problématique des feux de forêt. Ce constat nous autorise à penser que VOX est un protocole de routage optimisé dans un contexte de feux de forêt. Dans le chapitre suivant, nous proposons des résultats de simulation à partir de **DEVS-WSN** en focalisant sur l'étude des protocoles de routage et l'analyse de stratégies de déploiements déterministes de réseaux de capteurs sans fil.

CHAPITRE 5

Implémentation de DEVS-WSN et Résultats de simulation

La science, dans ses résultats, est plus magique que la magie : c'est une magie à preuves !

Extrait de La Carte d'identité, *Jean-Marie Adiaffi*

NOus allons aborder la partie implémentation et simulation de cette étude. Ce chapitre s'organise de la manière suivante. Nous commencerons dans une première partie en présentant la structure de l'application **DEVS-WSN**. Cette application est divisée en quatre paquetages dans un souci de modularité, "réutilisabilité" et de "modifiabilité". Ensuite nous proposons dans une seconde partie la définition du réseau simulé. La troisième partie introduit l'analyse des types de données reçus par la station de base. Ensuite dans une quatrième partie, nous comparons les résultats de simulation des trois protocoles de routage, GBR, Xmesh, notre algorithme VOX, en fonction de deux caractéristiques : la capacité énergétique et l'activité du processeur. Nous proposons dans la cinquième partie une étude de différentes stratégies de déploiement des réseaux de capteurs sans fil dans la problématique des feux de forêt. Enfin, dans une cinquième partie nous concluons ce chapitre.

5.1 Implémentation de DEVS-WSN

Les diagrammes de classes représentent un ensemble de classes/relations permettant une vue statique de la conception de systèmes logiciels [Capocchi, 2005]. L'implémentation DEVS-WSN résulte des modèles proposés dans le **Chapitre 3**. Le diagramme de classes que nous proposons est constitué de trois paquetages nommés ***DEVSModels***, ***WirelessSensorNetwork_Specification*** et ***Component Library*** liés par des relations de généralisation/spécialisation entre classes contenues. La Figure 5.1 présente le diagramme de classe gobale de l'application **DEVS-WSN** que nous avons développée. Nous n'y rendons visible que les attributs et les méthodes les plus importants.

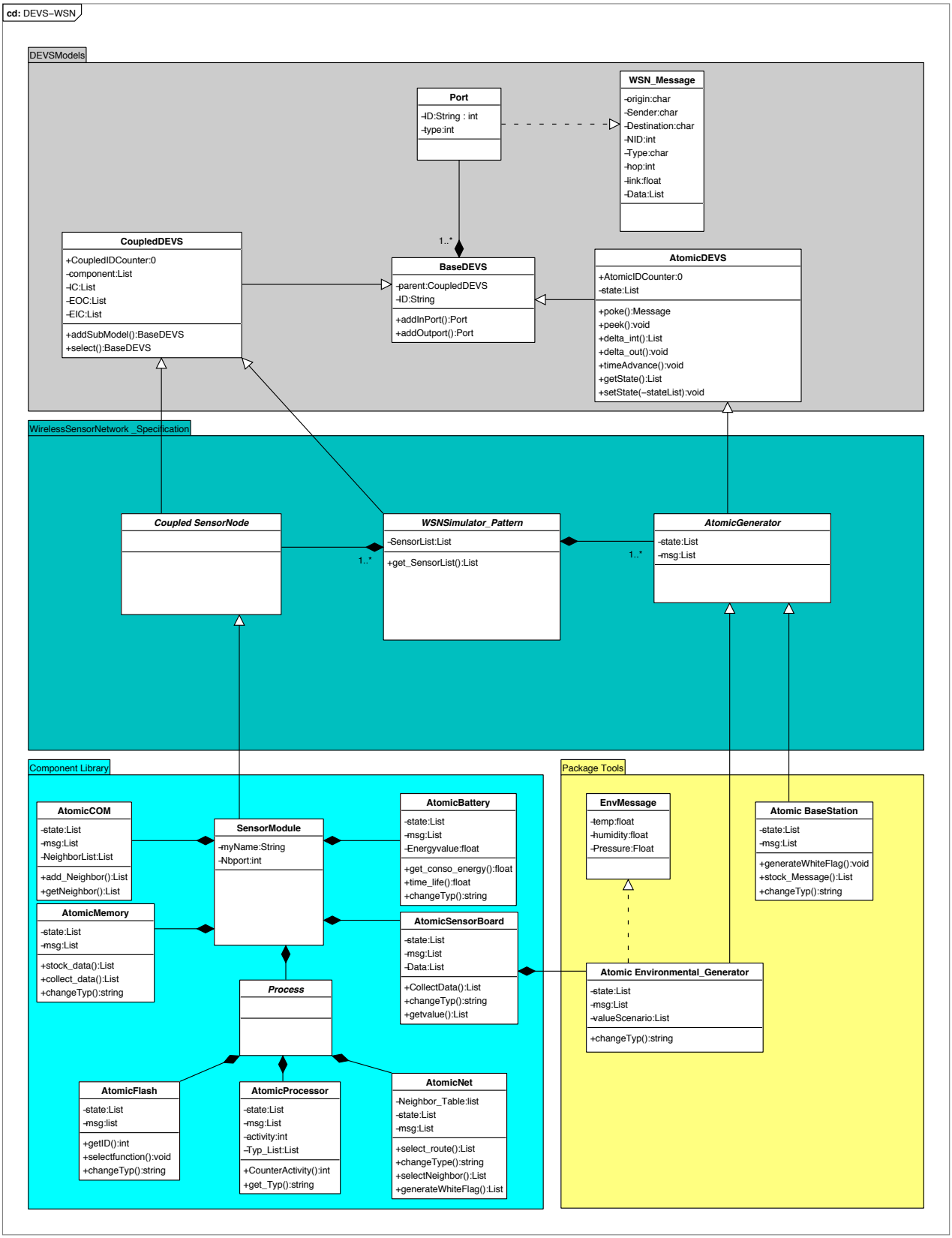


FIG. 5.1*: Diagramme de classe globale de l'application* ***DEVS-WSN***

5.1.1 Le paquetage *DEVSModels*

Le paquetage ***DEVSModels*** de [Bolduc et Vangheluwe, 2001] est un simulateur en python de modèles DEVS [Swinnen, 2003, Downey et al., 2002]. Nous utilisons ce package car nous l'avons expérimenté dans d'autres domaines, notamment en Bioinformatique[Antoine-Santoni et al., 2007a], et il s'avère complet pour la simulation de systèmes complexes.

L'arborescence des classes mises en oeuvre suit la philosophie DEVS selon laquelle :

- **l'aspect comportemental** d'un système est décrit par l'intermédiaire d'un ensemble de fonctions de transition au sein d'un modèle atomique ;
- **l'aspect structurel** est décrit par la définition des interconnexions entre modèles atomiques et/ou couplés au sein d'un modèle couplé.

Le paquetage ***DEVSModels*** de la Figure 5.1 est composé des deux classes abstraites "*Atomic-DEVS*" et *"CoupledDEVS*" héritant des propriétés de la classe "*BaseDEVS*". Elles permettent d'implémenter les composants atomiques et couplés DEVS. Les instances de la classe "*Port*" sont utilisées pour représenter les ports des composants DEVS. C'est grâce à l'interconnexion des composants que la structure du domaine peut se voir définie. La classe "*Message*" permet donc l'activation et l'échange d'informations entre les composants et est utilisée par la classe "*Port*".

Selon la philosophie de DEVS, il est naturel de considérer l'ensemble des modèles atomiques comme faisant partie d'un domaine comportemental et l'ensemble des modèles couplés comme appartenant à un domaine structurel. Ces deux domaines sont représentés par les deux classes abstraites.

5.1.2 Le paquetage "*WirelessSensorNetwork_Specifications*"

Les classes appartenant au paquetage ***WirelessSensorNetwork_Specifications*** constituent la base de notre architecture. La classe principale "*WSNSimulator_pattern*" hérite des propriétés d'un "*CoupledDEVS*" et constitue le modèle couplé principal du domaine étudié (*le plus haut niveau de description*). Il ne peut exister qu'une seule instance du *"WSNSimulator_pattern*". La

méthode get_SensorList permet de référencer les modèles de capteurs présents dans le réseau. La classe "*WSNSimulator_pattern*" va définir l'architecture du réseau en faisant intervenir deux classes : la classe "*Coupled SensorNode*" et la classe "*AtomicGenerator*". La classe "*Coupled SensorNode*" sera instanciée autant de fois que le réseau nécessitera des entités. La classe "*AtomicGenerator*" sera instanciée autant de fois qu'il y aura besoin de station réceptrice.

5.1.3 Le paquetage "*ComponentsLibrary*"

Le paquetage ***Component Library*** contient les composants nécessaires à la structure d'un capteur. Il est constitué par de classes héritant des propriétés d'un "*AtomicDEVS*" : "*AtomicCOM*", "*AtomicProcessor*", "*AtomicMemory*", "*AtomicBattery*", "*AtomicSensorboard*", "*AtomicNet*", "*AtomicFlash*" . Chaque classe aura la possiblité, grâce à la méthode get_message (non représentée sur Figure 5.1), de travailler sur la nature des données de ce dernier.

La classe "*AtomicCOM*" possède la méthode getNeighbor, qui lui permet, en fonction des caractéristiques de routage, d'orienter un message sur un port de sortie. La sortie dépendra de la correspondance avec la neighborList. La classe "*AtomicProcessor*" va permettre de traiter les données du message, en effectuant comme principale tâche l'analyse du typage par le biais de la méthode get_Typ. La classe "*AtomicNet*", par le biais de sa méthode select_route, fournit au message ces informations de routage en fonction du protocole de routage défini. La classe "*AtomicFlash*" aura le but, par le biais de sa méthode getID, de récupérer par exemple l'identificateur de groupe ID. Par le biais de la méthode changeTyp, la classe "*AtomicFlash*" a la possibilité d'analyser le type de message et de modifer certaines valeurs. On note la présence également d'une classe abstraite héritant des propriétés d'un "*CoupledDEVS*" : "*Process*" permettant de faire le couplage entre les trois classes : *"AtomicProcessor*" "*AtomicNet*", et"*AtomicFlash*" . La classe "*AtomicBattery*" a pour objectif de représenter la gestion de la consommation énergétique dans le capteur par le bais de la méthode get_conso_energy. La classe "*SensorModule*" va représenter le couplage général de tous les composants. Il représente la structure statique du capteur. La classe "*SensorNode*" hérite des caractéristiques d'un "*CoupledDEVS*" et de la classe "*SensorModule*" , qui va permettre de différencier les noeuds à travers le réseau par le biais d'un nom.

5.1.4 Le paquetage *Tools*.

Ce paquetage va regrouper tous les outils nécessaires pour compléter l'architecture d'un réseau de capteurs.

Nous y trouvons 2 types de classes. La première,"*AtomicBaseStation*", va correspondre à la station de base dans un réseau. Elle possède deux méthodes : la méthode generateWhiteFlag qui va lancer périodiquement un message de synchronisation du système ; et la méthode stock_Message, qui a pour but de collecter les informations nécessaires qui vont permettre l'analyse de l'environnement. Cette méthode va stocker toutes les informations dans un tableau. La seconde classe est *"AtomicEnv*" qui va permettre la description des modèles environnementaux. Elle est intimement liée à la classe"*AtomicSensorBoard*". Cette classe correspond bien à un outil permettant d'affiner le modèle de capteur. Par conséquent, d'autres outils peuvent être rajoutés dans le paquetage ***Tools*** comme des déclencheurs par exemple en effectuant une action sur le réseau. Cette classe pourra transmettre un message environnemental "*EnvMessage*" à la classe "*AtomicSensorBoard*".

Nous venons de définir la structure de l'application **DEVS-WSN**, architecture modulaire basée sur le simulateur DEVS de [Bolduc et Vangheluwe, 2001]. Nous proposons la description d'un premier réseau qui va être soumis à une première série de simulation [Antoine-Santoni et al., 2007b, Antoine-Santoni et al., 2007d].

5.2 Définition du réseau simulé

Dans notre premier cadre de simulation, nous proposons un réseau constitué de huit noeuds. Ce réseau est limité par certaines contraintes limitant les communications :

1. la présence d'éléments géographiques ou matériels empêchant les communications (murs, rochers,...) ;
2. l'éloignement trop important entre les noeuds pour une communication radio.

Ces deux contraintes physiques sont communes et sont des facteurs influençant de manière importante le réseau. La Figure 5.2 représente un réseau de 8 noeuds avec une station de base disposée

de manière à tenir compte des contraintes physiques rencontrées. La possibilité que deux noeuds ne puissent pas communiquer entre eux est représentée dans l'arbre de simulation DEVS (*cf* 3.2.2) par une absence de couplage des MC "*Sensor*".

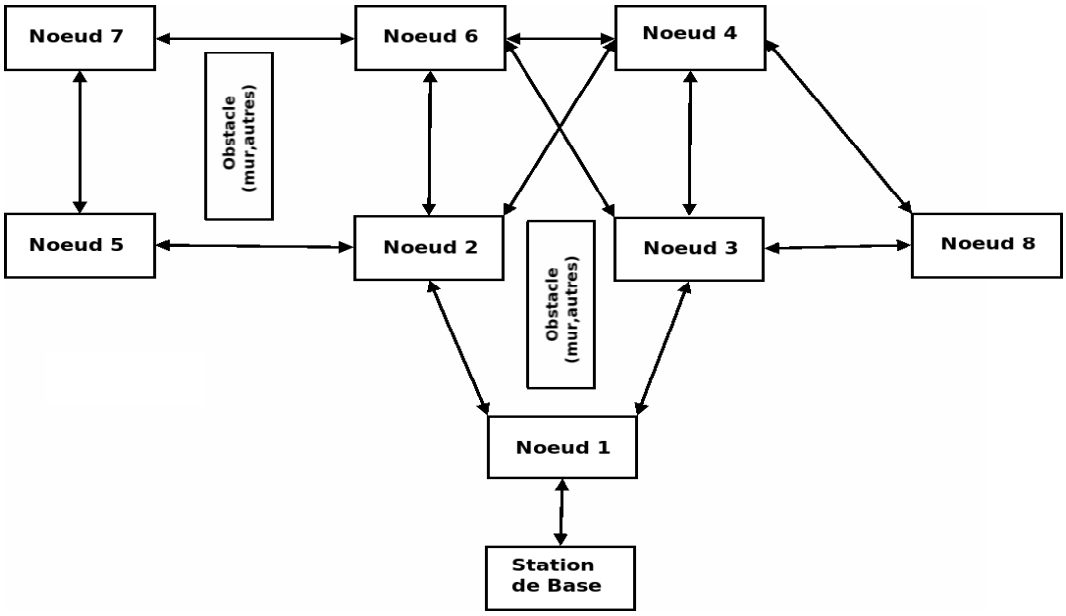


FIG. 5.2*: Structure du réseau à 8 noeuds*

La définition du réseau, que nous soumettons à la simulation, va mettre en concurrence des noeuds pour étudier les protocoles de routage. Nous représentons dans le Tableau 5.1, les tables de voisinage :

Noeud	Voisin 1	Voisin 2	Voisin 3	Voisin 4
Station de Base (BS)	Noeud 1			
Noeud 1	BS	Noeud 2	Noeud 3	
Noeud 2	Noeud 1	Noeud 4	Noeud 6	Noeud 5
Noeud 3	Noeud 1	Noeud 4	Noeud 6	Noeud 8
Noeud 4	Noeud 2	Noeud 3	Noeud 8	Noeud 6
Noeud 5	Noeud 2	Noeud 7		
Noeud 6	Noeud 2	Noeud 3	Noeud 4	Noeud 7
Noeud 7	Noeud 6	Noeud 5		
Noeud 8	Noeud 3	Noeud 6		

TAB. 5.1*: Tableau des voisinages de chaque noeud selon la Figure 5.2*

Les tables de voisinage relatives à chaque noeud seront identiques dans le contenu à ce qui est présenté dans le Tableau 5.1. La classification de cette table sera réalisée au fur et à mesure de la simulation par le protocole de routage et en particulier par son algorithme de sélection du voisin idéal.

Les données arrivant sur la station de base seront celles dans un premier temps qui seront analysées. Le Tableau 5.2 représente un exemple de résultats que l'on peut obtenir :

Relais	Type	Parent	Origine	Temps	Activité	Temp	Conso
Noeud 1	ACK	BS	Noeud 1	0,4	5	12	2995
Noeud 1	ACK	Noeud 1	Noeud 2	0,8	5	12,2	2997
Noeud 1	ACK	Noeud 1	Noeud 3	1,2	5	12,1	2999
Noeud 1	ACK	Noeud 2	Noeud 4	1,6	5	11,9	2998

TAB. 5.2*: Exemple de tableau de résultats*

Dans le Tableau 5.2, nous retrouvons certains champs qui constituent la base des messages échangés dans le réseau (*cf* 4.1.1). Le champ **Relais** désigne ici quel est le noeud qui a envoyé ce message. Nous remarquons dans l'exemple que seul le Noeud 1 communique naturellement car c'est le seul qui a cette possibilité. Le champ **Type** de message qui correspond majoritairement au type *ACK*. Cependant le type *WhiteFlag* sera présent, correspondant à l'envoi de messages de mise à jour du Noeud 1 avec la station de base. Le champ **Origine** définit le noeud créateur de ce message. Le champ **Temps** correspond aux temps d'arrivée de chaque message sur la station de base. Les champs **Activité**, **Temp** et **Conso** sont les données enregistrées à l'envoi du message. Nous allons présenter l'analyse des résultats en distinguant les informations communes à tous les protocoles et les informations permettant une comparaison pertinente.

5.3 Analyse des données reçues sur la station de base

Nous présentons dans cette partie, les données réceptionnées par la station de base. Nous pouvons, grâce au modèle de capteur développé et au type de message envoyé par les noeud,

disposer de plusieurs types de données. Nous étudierons les trois protocoles de routage GBR, Xmesh et notre protocole VOX (*cf* 4.3).

5.3.1 Caractéristiques communes à tous les protocoles

5.3.1.1 Le nombre de messages reçus

Nous analysons le nombre de message reçus par la station de base après 20 minutes de simulation. Ce nombre est commun à tous les protocoles étudiés. Nous voyons sur la Figure 5.3 que ce nombre est constant et croît naturellement au cours du temps.

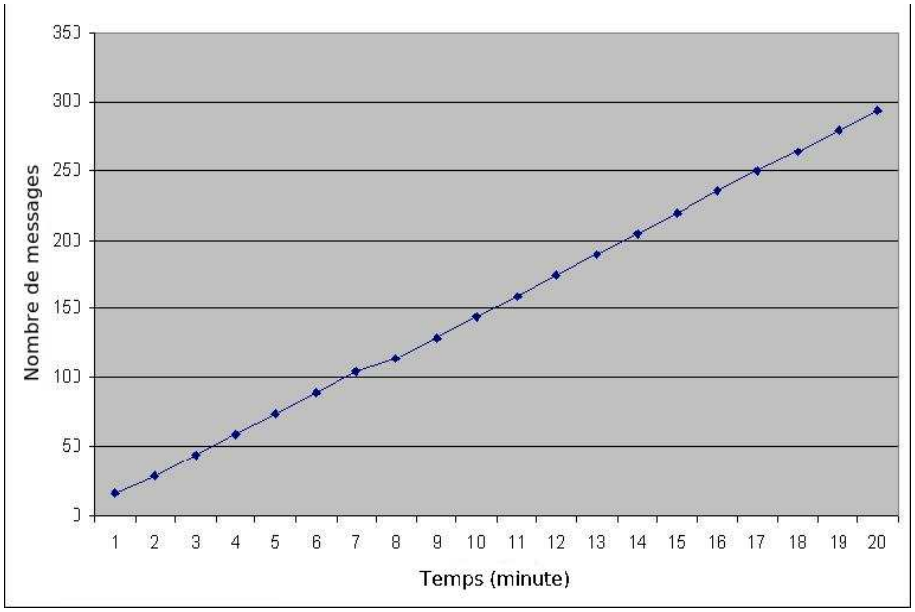


FIG. 5.3*: Nombre de messages arrivant sur la station de base*

Le nombre de messages arrivant sur la station de base est important. Sur la Figure 5.3, nous pouvons voir la quantité de messages arrivant sur la sation de base. En effet, les capteurs transmettent leurs informations les uns après les autres. Il est important de préciser que seuls les messages qui arrivent à la station de base sont référencés. Nous ne faisons pas apparaître les messages échangés entre les noeuds. La valeur réelle des messages échangés dans le réseau durant 20 minutes de simulation sur le réseau à huit noeuds est proche de 2000 messages.

Pour affiner ce premier type de résultats, nous proposons de visualiser également le nombre de messages reçus en fonction de chaque noeud du réseau, illustré par la Figure 5.4. Ceci est permis par le champ ***Relais*** du message, qui précise le noeud générateur du message.

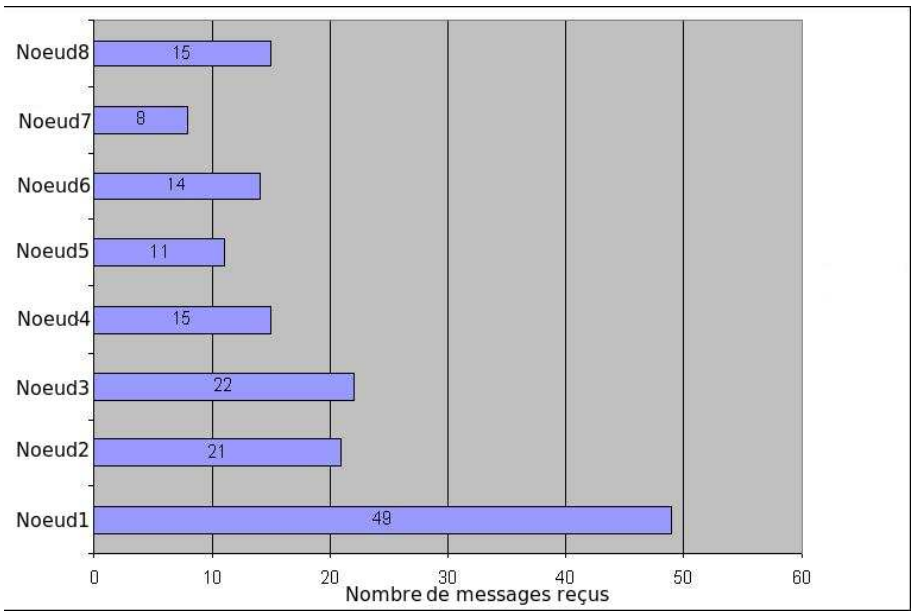


FIG. 5.4*: Nombre de messages arrivant sur BS en fonction de l'origine pour 10 mn de simulation*

Ces résultats sont directement dépendants de la structure de notre réseau à huit noeuds. Nous distinguons l'activité intensive du Noeud 1. En effet, ce dernier est dans une position centrale et possède une activité importante. Nous voyons qu'il génère plus de messages. Nous devons faire une précision sur ce fait. En effet, le Noeud 1 posséde une relation particulière avec la station de base, car elle est dans son voisinage. C'est pour cela qu'un quart des messages appararaissant sur la station de base sont de type"*WhiteFlag*", messages de mise à jour. En effet, le Noeud 1 considère naturellement que la station de base est un noeud comme les autres et demande une synchronisation par le biais d'un message de type "*WhiteFlag*". Pour suivre ce processus, nous faisons le choix de ne pas filtrer ces messages et de les comptabiliser dans les résultats finaux.

Un autre aspect important est la différence entre le nombre de messages en fonction de chaque noeud. Nous observons que la station de base a reçu moins de messages du Noeud 7 que du Noeud 2 par exemple. Le Noeud 7 est le noeud qui doit réaliser le nombre de sauts le plus important pour

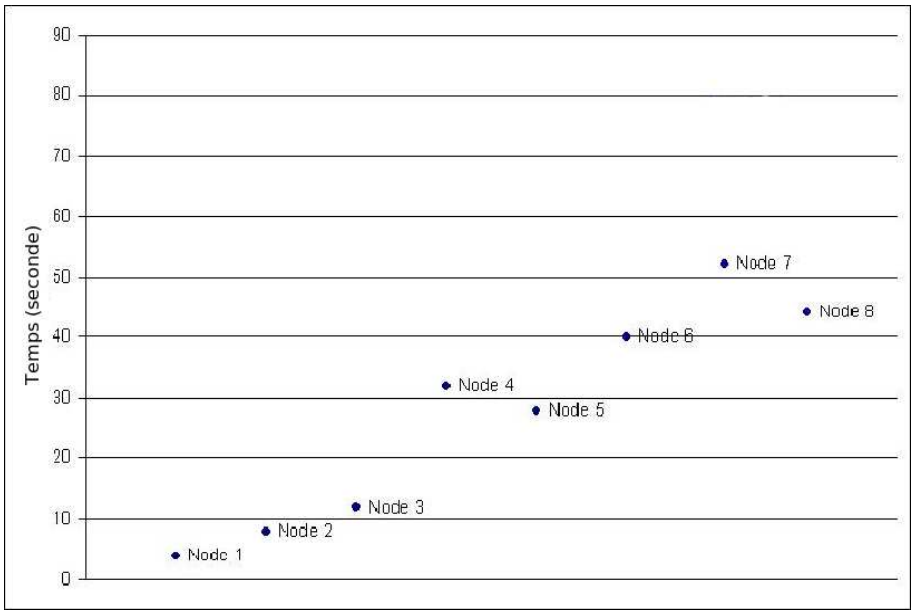


FIG. 5.5*: Temps d'arrivée des messages en fonction de l'origine*

atteindre la station de base et par conséquent ces messages metttent plus de temps pour arriver et ne sont donc pas comptabilisés à l'instant t.

5.3.1.2 Temps d'arrivée des messages sur la station de base

Cela se confirme par la Figure 5.5. En effet, nous proposons les temps d'arrivée de chaque premier message en fonction de son origine. Nous voyons que le Noeud 1 est le premier à répondre à la sollicitation de la station de base. Nous voyons surtout que le Noeud 7 apparaît dans la station de base et ce au bout de 52 secondes de simulation.

Il est important de préciser que cet intervalle doit être relativisé dans sa valeur. En effet, la mise en place du réseau (la phase de création du gradient) nécessite un échange important de messages. Cela crée un déséquilibre au regard des temps d'apparition des messages. Cela apparaît clairement sur la Figure 5.5, où l'on peut distinguer de manière assez simple quels sont les noeuds les plus proches de la station de base et ceux plus éloignés, et ce qui permet de déterminer la structure du réseau. Cet intervalle de temps assez conséquent pour le Noeud 7 va diminuer au cours de la simulation pour faire apparaître des messages dans la station de base, environ toutes les 40 secondes. Ce décalage dans le temps nous permet de comprendre que le faible nombre de messages relatifs au Noeud 7 provient uniquement du fait que ses messages sont en cours de transit

dans le réseau. Les noeuds qui servent de relais aux messages des noeuds les plus éloignés sont soumis à d'autres messages issus de leur voisinage direct et qui ont, dans la phase d'initilisation, ralenti le transfert d'informations.

Nous rappelons que la structure du protocole VOX diffère dans la gestion des files d'attente du protocole Xmesh. Nous précisons que cette modification n'améliore pas les performances du protocole pour l'envoi des messages. Cependant l'amélioration ne crée aucun conflit dans la gestion des messages, elle a permis seulement une simplification dans l'implémentation du protocole dans le MA "*Net*".

5.3.2 Comparaison de protocoles de routage

Nous proposons dans cette partie, une comparaison de trois protocoles GBR, Xmesh et notre protocole VOX. Pour analyser de manière judicieuse les performances des différents algorithmes de routage, nous allons utiliser la consommation énergétique et l'activité du modèle processeur comme paramètres de comparaison.

5.3.2.1 Consommation énergétique

Les résultats de simulation sont basés sur un réseau à 8 noeuds illustré par la Figure 5.2. Nous proposons d'étudier la consommation énergétique pour chaque noeud durant 20 minutes de simulation. Cette consommation est représentée par la diminution de la capacité énergétique au cours du temps pour chaque noeud. Sur la Figure 5.6, nous représentons les résultats après simulation. La Figure 5.6 a) est la consommation énergétique sur un réseau utilisant le protocole *GBR*, la Figure 5.6 b) concerne le réseau utilisant le protocole *Xmesh* , la Figure 5.6 c) illustre notre prototype de protocole dérivé de Xmesh, VOX.

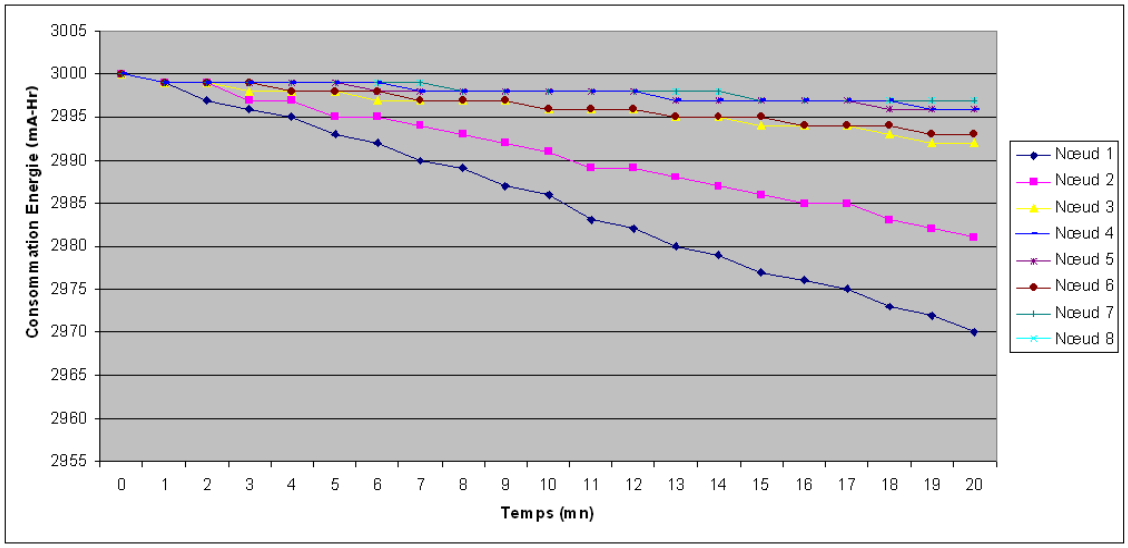
La comparaison globale des trois protocoles nous fait remarquer l'importante consommation du Noeud 1. Le Noeud 1 est le point relais entre la station de base et tous les autres noeuds. Il reçoit les informations de tous les autres noeuds et ce qui justifie son importante activité de ce noeud dans le réseau . Il est important de préciser que le protocole de routage a peu d'importance pour le Noeud 1. En effet ce dernier possède une relation privilégiée avec la station qui représentera

naturellement toujours le voisin idéal étant donné que le nombre de sauts est égal à 0.

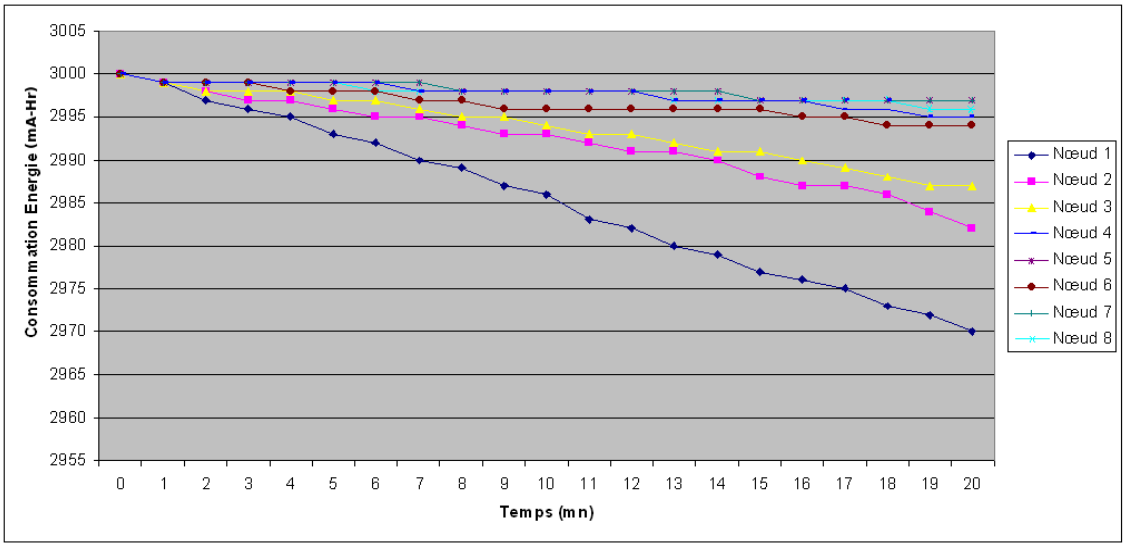
La consommation énergétique va varier de manière conséquente en fonction du protocole de routage. Nous focaliserons notre analyse sur les noeuds qui peuvent avoir un rôle de routage, c'est à dire les noeuds 2, 3, 5 et 6.

La Figure 5.6 a) concerne le réseau utilisant le protocole GBR ; nous voyons clairement apparaître une importante consommation du Noeud 2. Nous rappelons que dans la sélection du voisin idéal seul le nombre de sauts (métrique 1) est pris en compte et qu'à niveau équivalent, un choix aléatoire est fait entre les noeuds. Nous nous devons de préciser que la Figure 5.6 a) ne résulte pas de la première simulation réalisée avec ce protocole mais le résultat d'une suite de simulations. Lors d'un cas précis de simulation, nous avons observé un décalage important dans la consommation des noeuds conséquence directe conséquence de l'algorithme de routage. Les noeuds 2 et 3 sont soumis à une concurrence importante vis à vis des noeuds 4 et 6. Lors de la simulation, la fonction aléatoire a choisi de manière importante de fixer le Noeud 2 comme voisin idéal dans les tables de routage des noeuds 6 et 4. Cela se traduit par une surconsommation du Noeud 2 vis à vis du Noeud 3 qui possède pourtant le même nombre de sauts. Le Noeud 3 rejoint, au niveau de la consommation, le Noeud 6. Le Noeud 6 posséde une consommation importante relative à sa mise en concurrence avec le Noeud 5 dans la table de routage du Noeud 7. Nous pouvons apercevoir un léger décalage entre la consommation du Noeud 5 et celle du Noeud 6.

a)



b)



c)

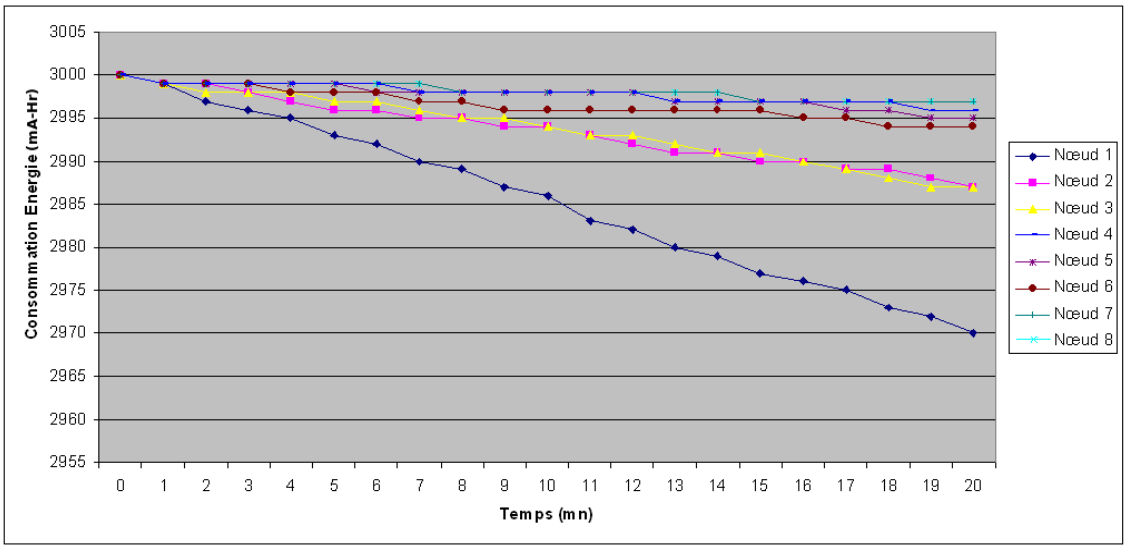


FIG. 5.6*: Consommation énergétique de chaque noeud en fonction des trois protocoles*

Dans ce cas précis, il semblerait que la sélection réalisée dans la table de routage du Noeud 7 se soit faite de manière assez fréquente aux dépends du Noeud 5, augmentant par conséquent la consommation du Noeud 6.

La Figure 5.6 b) nous représente également la consommation énergétique de tous les noeuds implémentés avec le protocole Xmesh. Ce graphique est issu également d'une série de simulations. Nous observons que les courbes se sont harmonisées de manière plus importante entre les noeuds soumis à concurrence dans le routage par rapport à la Figure 5.6 a). Nous distinguons une régression du déséquilibre entre la consomnation du Noeud 2 et 3.

L'introduction d'une deuxième métrique (qualité des liens) réorganise la structure des tables de routage des noeuds 6 et 4. Nous voyons un resserrement des courbes de consommation énergétique. Ceci est également observable entre les noeuds 5 et 6 dont les courbes de consommation se superposent. Cet aspect se traduit par le protocole lui-même. Le Noeud 6 est soumis à plus importante consommation étant donné le nombre de voisins qu'il possède. Cependant la meilleure qualité des liens entre le Noeud 5 et le Noeud 7 favorise les communications entre ces derniers. Le Noeud 6 ne possède que très rarement une action de routage vis à vis du Noeud 7. Ce partage des tâches entre les noeuds 5 et 6 permet d'avoir un équilibre sur la consommation. La deuxième métrique a permis d'équilibrer les consommations énergétiques. Précisons que la nature même de Xmesh ne vise pas à une économie d'énergie mais se focalise sur la qualité de la transmission. Cependant dans ce cas précis, résultat d'une série de simulations, nous voyons se réaliser une économie d'énergie insoupçonnable.

Nous pouvons à présent visualiser la Figure 5.6 c) avec l'ajout d'une troisième métrique se basant sur la capacité énergétique de l'algorithme VOX. Concernant les noeuds 2 et 3 soumis à concurrence dans les tables de routage des noeuds 4 et 6, la première observation concerne l'équilibre qui s'opère entre les deux noeuds du réseau. Nous voyons que les noeuds tendent à ne pas maintenir une consommation excessive pour un noeud. Nous observons un croisement perpétuel des deux courbes. Le rééquilibrage qui s'opère est expliqué par l'ajout de la troisième métrique. En effet, l'algorithme VOX propose de favoriser un noeud avec une plus grande capacité énergétique si ce dernier a une qualité de liens inférieure à 5% par rapport au noeud concurrent.

Même si durant un certain temps de simulation, l'écart est important, nous voyons qu'à terme les courbes de consommation se rejoignent. Les noeuds 5 et 6 proposent le même type de courbe. Nous rappelons que les noeuds 5 et 6 sont concurrents dans la table de routage du Noeud 7. Le protocole VOX va permettre un partage de la consommation énergétique pour une qualité de lien équivalente. Cependant le Noeud 6 possède un voisinage plus conséquent que le Noeud 5, ce qui augmente sa consommation d'énergie. Ceci se confirme sur plusieurs simulations. Pour faire une comparaison entre les noeuds 5 et 6, le quasi-équilibre trouvé par Xmesh est le fruit d'un seul cas de simulation, contrairement à VOX qui mantient cette particularité de simulation durant un certain nombre de simulations.

Dans tous les cas présentés ici nous avons peu parlé des noeuds 4, 7 et 8. En effet ces noeuds ne sont pas soumis au rôle de routeur et par conséquent leur consommation ne réflètent pas une véritable action de routage ou de réelle mise en concurrence dans les tables de routage. La diminution de leur capacité énérgétique va être relative à leur position dans le réseau, que nous résumerons en indiquant qu'un noeud sans sollicitation pour le routage, aura une consommation relative à l'activité de communication avec son voisinage direct.

Nous venons de voir que le protocole VOX répartissait, de manière plus efficace, les tâches de rouatge entre les noeuds. Ce meilleur équilibre dans les rôles de chaque noeud permet de réduire le risque d'avoir des noeuds en surconsommation d'énergie. Nous savons que le modèle énergétique s'appuie sur un modèle linéaire de consommation énergétique(*cf* 4.1.5.1), et qu'il existe un modèle prenant en compte la relaxation de la batterie. Pour permettre une visualisation supplémentaire de l'activité d'un noeud, si l'on considère que le modèle linéaire ne reflète pas complètement la réalité, nous proposons de comparer les protocoles en fonction de l'activité du processeur.

5.3.2.2 Activité du processeur

Pour affiner les résulats précédents, nous présentons l'activité du processeur de chaque noeud en fonction des protocoles de routage. L'activité du processeur ne représente en aucun cas le nombre de messages envoyés, mais les actions réalisées par le MA "*Processor*". Cette valeur

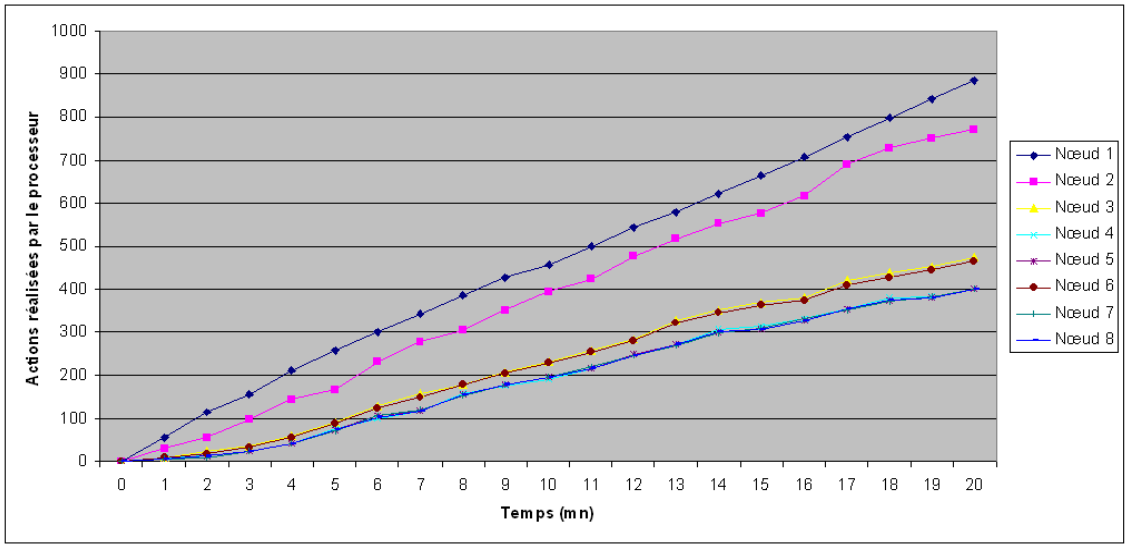
est la somme des activités de communication classiques et des activités de routage. L'activité du processeur décrit le rôle de chacun des noeuds durant la simulation.

Les Figures 5.7 a), 5.7 b) et 5.7 c) représentent respectivement les activités des noeuds selon les protocoles GBR, Xmesh et VOX.

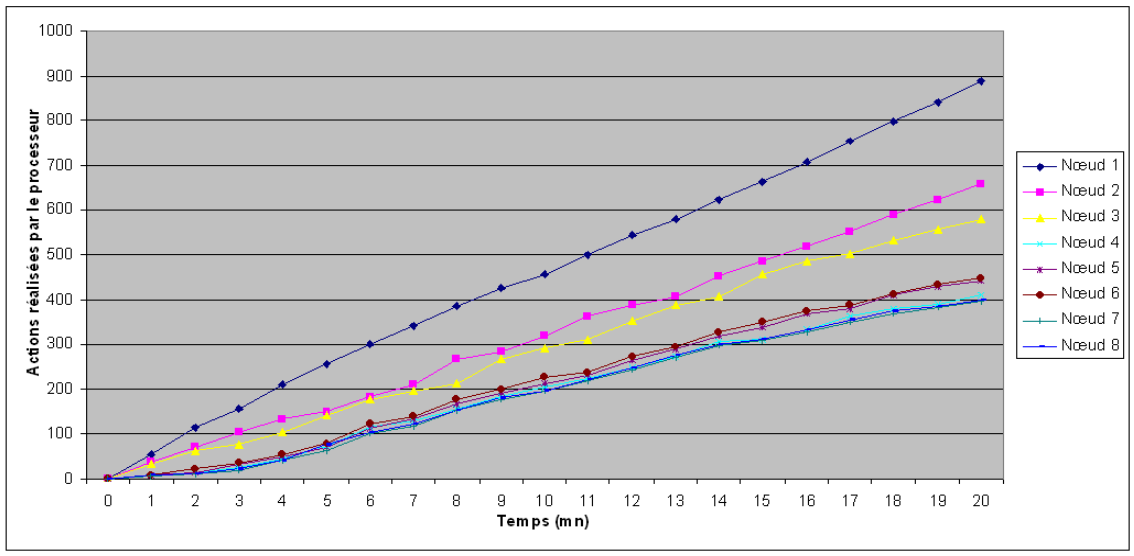
La première analyse confirme l'extrême activité du Noeud 1. Conformément à ce qui a été observé avec la consommation énergétique, ce noeud possède une position centrale dans le réseau, collectant et relayant énormément de messages. Nous focalisons notre analyse sur les résulats des noeuds 2, 3, 5 et 6.

Nous voyons que les noeuds avec une consommation importante d'énergie possèdent naturellement une forte activité au niveau du processeur. Le protocole GBR propose le même déséquilibre au niveau des noeuds 2 et 3. L' ajout de la métrique 2 dans le protocole Xmesh réduit l'activité du Noeud 2 tout en augmentant l'activité du Noeud 3, en essayant de répartir les tâches. L'algorithme de routage VOX confirme la tendance au partage complet des tâches sur l'ensemble de la simulation entre le Noeud 2 et 3. Concernant les noeuds 5 et 6, ces résultats valident la première analyse des capacités des protocoles de routage en reconnaissant les mêmes particularités observables avec la consommation énergétique. En effet, le protocole GBR par son processus aléatoire de sélection du voisinage, déséquilibre de manière importante l'activité de certains noeuds. La tendance au déséquilibre est corrigée de manière conséquente avec le protocole Xmesh, même si le cas de simulation présenté par la Figure 5.7 b) est exceptionnel. Notre protocole VOX oriente le réseau vers un équilibre plus conséquent. Nous observons que les équilibres, relevés sur la consommation énergétique, sont confirmés par l'activité du processeur. Le résultats présentés viennent valider les résultats de consommation. En effet, les noeuds soumis au rôle de routeur, après discrimation par les différents algorithmes de routage, possédent une activité conséquente. L'algorithme de sélection est un facteur déterminant dans le partage des tâches. Le protocole GBR propose un routage de l'information mais ne tient compte ni du maintien du niveau énergétique ni d'un réel partage des tâches pour des noeuds soumis à concurrence dans les tables de routage.

a)



b)



c)

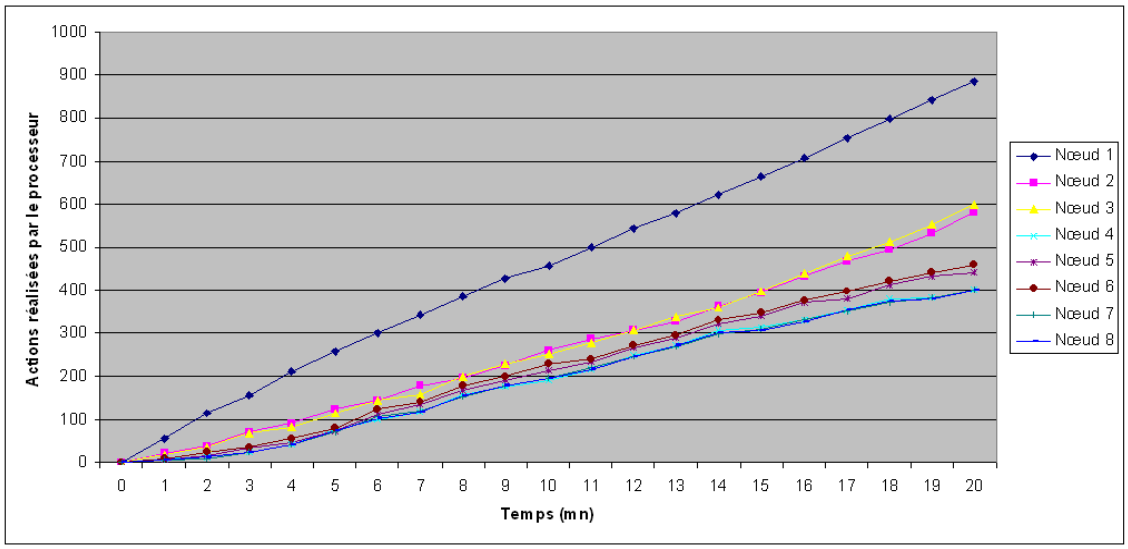


FIG. 5.7: *Activité du processeur de chaque noeud en fonction des trois protocoles*

Le protocole Xmesh favorise un retour à l'équilibre mais nous démontrons que ce protocole peut être pris en défaut. Malgré l'ajout d'une deuxième métrique par rapport au protocole GBR, il ne guarantit que très rarement un réel partage efficace des activités des noeuds. L'algorithme VOX, que nous avons proposé, rectifie les phénomènes de disparité observés avec le protocole GBR et le protocole Xmesh, en favorisant une réelle alternance dans les rôles de chaque noeud. Cette amélioration est la conséquence directe de l'ajout des métriques 2' et 3 décrites par les conditions 1 et 2 basés sur la consommation énergétique.

Nous avons vu que le protocole VOX s'avère plus efficace dans la répartition des tâches de routage entre les noeuds du réseau. Ce fait se confirme en visualisant la fréquence d'élection du voisin idéal pour les trois protocoles. Une fréquence d'élection réguilière montre la capcité du protocole à répartir les tâches des noeuds du réseau.

5.3.2.3 Fréquence d'élection du voisin idéal

Pour compléter les résultats précédents, nous présentons dans cette partie la fréquence de changement du voisin idéal dans les tables de routage. Nous rappelons que la sélection du voisin idéal se fait selon les algorithmes de routage. Le voisin idéal va correspondre au premier noeud dans la table de routage lors des différents instants de simulation. Nous n'obtenons pas ces informations sur la station de base, c'est le fruit d'une analyse durant la simulation. La Figure 5.8 nous présente les résultats des différentes élections de premier voisin pour le Noeud 4 en fonction des différents protocoles de routage. Les deux noeuds mis en concurrence sont le Noeud 2 et le Noeud 3. Nous précisons que ces changements sont pris en compte à un instant *t* et ne représentent pas la totalité des changements.

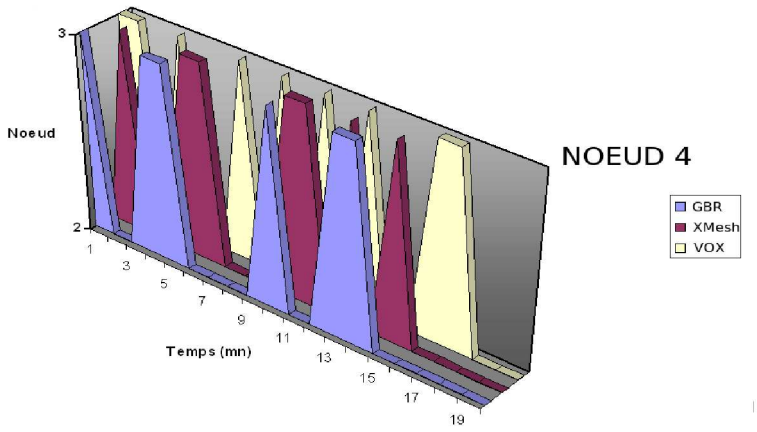


FIG. 5.8*: Fréquence d'élection du voisin idéal pour le Noeud 4*

La Figure 5.8 confirme les résultats exprimés jusqu'à présent. En effet, nous voyons que l'alternance entre les noeuds pour le protocole GBR est faible. Ceci est la conséquence de l'utilisation d'une seule métrique pour la définition du voisin idéal et de la forte dépendance à la fonction aléatoire qui détermine, à niveau de sauts équivalents, le voisin idéal. La fréquence de changements augmente de manière assez distincte pour le protocole Xmesh. En effet, il permet grâce à sa deuxième métrique de favoriser le partage des rôles entre les Noeuds 2 et 3 même si ce n'est pas le rôle essentiel de ce dernier. L'amélioration notable est réellement significative avec l'algorithme VOX qui permet une alternance dans la sélection du voisin idéal dans la table de routage du Noeud 4. L'ajout de la troisième métrique et la redéfinition de la sélection des noeuds en fonction de la quantité des liens favorise un rééquilibrage des tâches dans le réseau. Les résultats présentés pour le Noeud 4 sont de même nature pour les noeuds 5 et 6 que nous présentons sur la Figure 5.9.

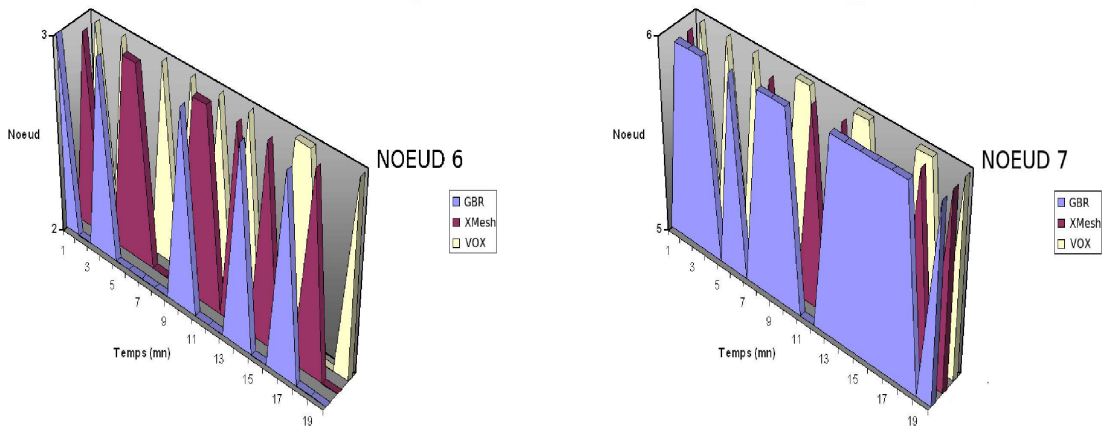


FIG. 5.9*: Fréquence d'élection du voisin idéal pour le Noeud 6 et 7*

Le Noeud 6 propose la mise en concurrence des noeuds 2 et 3 et le Noeud 7 effectue un processus de sélection entre le Noeud 5 et 6. Pour le table de routage des noeuds 6 et 7, l'algorithme VOX démontre une qualité certaine qui est de favoriser l'alternance dans le routage permettant par conséquent une véritable augmentation de la durée des entités du réseau.

5.3.3 Synthèse

L 'algorithme VOX basé sur la définition de trois métriques (nombre de sauts, qualité des liens et capacité énergétique) comme critère de sélection améliore de manière significative la durée de vie du réseau. Sans compromettre la définition du plus court chemin vers la station de base et la qualité de la transmission, il permet de répartir de manière plus équitable les tâches entre les noeuds. Les conditions 1 et 2 (*cf* 4.3.5) offrent un processus de sélection plus pertinent. En effet, nous rappelons que les protocoles GBR et Xmesh sont des protocoles dits " plats" , que nous avons identifiés comme les plus efficaces dans le contexte de notre étude. Le protocole VOX s'avère plus efficace que GBR et Xmesh, en offrant un meilleur partage des tâches dans le réseau entre les noeuds. La durée de vie est la période durant laquelle un réseau sera capable d'accomplir une tâche (collecte de données environnemenales) de la manière la plus efficace. Si durant une période, un noeud ne fonctionne plus, à cause par exemple du manque de ressources énergétiques, une partie du réseau sera défailante. Le protocole VOX, en favorisant un meilleur partage des tâches entre les noeuds et donc une meilleure répartition des consommations énergétiques, augmente la durée

de vie du réseau. De ces constats, l'algorithme VOX s'inscrit comme un protocole efficace dans la problématique des feux de forêt.

Les protocoles de routage permettent un aiguillage des informations vers la station de base. Cependant ces protocoles peuvent être influencés par le déploiement des capteurs. Nous proposons à présent un autre axe d'étude. Nous voulons étudier les stratégies de déploiement des réseaux de capteurs sans fil soumis à un feu de forêt.

5.4 Stratégies déterministes de déploiement

Le déploiement d'un réseau peut être déterministe ou aléatoire. Dans une disposition déterministe, la position des capteurs est connue à l'avance et est le résultat d'une mise en place manuelle. Un déploiement aléatoire peut consister à déployer un réseau de capteurs d'un avion au dessus d'une forêt. Par conséquent, la position des capteurs n'est pas connue immédiatement. Nous proposons, dans cette partie, d'étudier différentes dispositions de réseaux de capteurs et à les soumettre à un scénario de feux de forêt. Nous faisons d'abord un rappel du principe de détection environnementale qui est utilisé dans nos capteurs. Ensuite nous proposerons une étude théorique de différentes stratégies de déploiement et nous ferons enfin une synthèse.

5.4.1 Présentation

Le déploiement possède un rôle stratégique dans la transmission des données. En effet, une mauvaise disposition des capteurs peut entraîner un perte conséquente de données.

Pour étudier ce phénomène, nous implémentons le MA "*Env*" de chaque capteur avec un schéma simple de feu. Dans le cadre de notre application, nous avons vu que les résultats sont intreprétés sous la forme de tableau de messages reçus par la station de base. Nous avons constaté qu'il existe deux cas critiques : épuisement énergétique et augmentation importante de la température. Dans ces deux cas, le capteur passe dans un état ***'DEAD'***, état dans lequel le capteur ne peut plus recevoir de messages, ni en envoyer. Le scénario de "propagation de feu" que nous proposons consiste à choisir aléatoirement plusieurs noeuds dans le réseau et de les soumettre à des tempéra-

tures importantes, pour qu'ils passent d'un état actif '***BUSY***' à un état ***'DEAD***', correspondant en fait à une destruction de capteurs sur le passage d'un feu. Nous signifions à certains noeuds, par le biais du MA "*Env*", une valeur de température importante supérieure à un seuil arbitraire que nous fixons à 70°C. Lorsqu'un noeud est sollicité pour une collecte d'informations, ce dernier va solliciter le MA "*Sensorboard*" et par conséquent le MA"*Env*". Le modèle environnemental contient des valeurs critiques pour certains noeuds à différents temps de simulation : par exemple le Noeud 8, qui ira collecter des informations entre la 6ième minute et la 8ième minute, recevra une valeur de température supérieure au seuil. Au moment de la réception du message, MA "*Sensorboard*" va détecter la valeur critique. Il va envoyer un message de type *DEAD* au modèle MA "*Processor*" qui va ensuite arriver au MA "*COM*". Le Noeud 8 apparaît comme défaillant dans le réseau. Nous basons notre réflexion sur les stratégies de déploiement à partir de ce concept de défaillance.

Au niveau du protocole de routage, nous implémentons les noeuds avec le protocole Xmesh. Le protocole Xmesh est le protocole utilisé par les capteurs MICA2, plate-forme que nous étudions dans le chapitre suivant.

Pour une meilleure lisibilité, nous proposons les résultats sous la forme de graphes représentant d'abord le déploiement soumise à un test de défaillance (cas de destruction d'une noeud par le feu) et ensuite la représentation des résultats sur la station de base obtenus après analyse.

5.4.2 Déploiement simple en grille

Nous proposons d'étudier un premier cas de déploiement simple en grille à l'un d'un réseau composé de 60 capteurs et d'une station de base comme représenté sur la Figure 5.10.

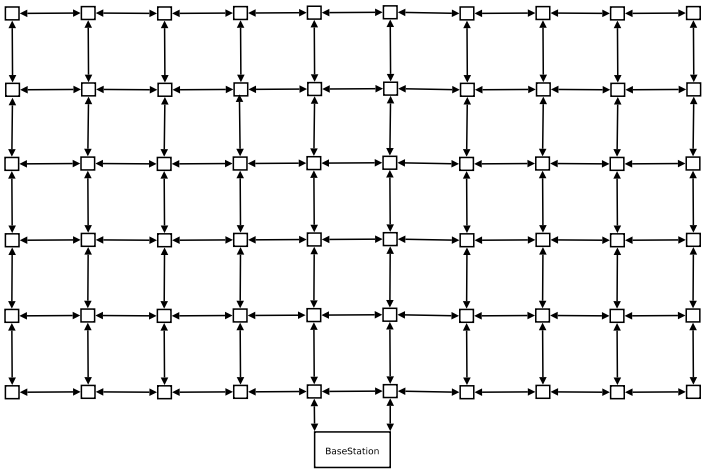


FIG. 5.10*: Déploiement simple en grille*

Sur le concept de défaillance du système énoncé précédemment, nous définissons les noeuds défaillants. Ces noeuds sont représentés sur la Figure 5.11 par une zone rouge, précisant de manière symbolique la position du feu sur le réseau.

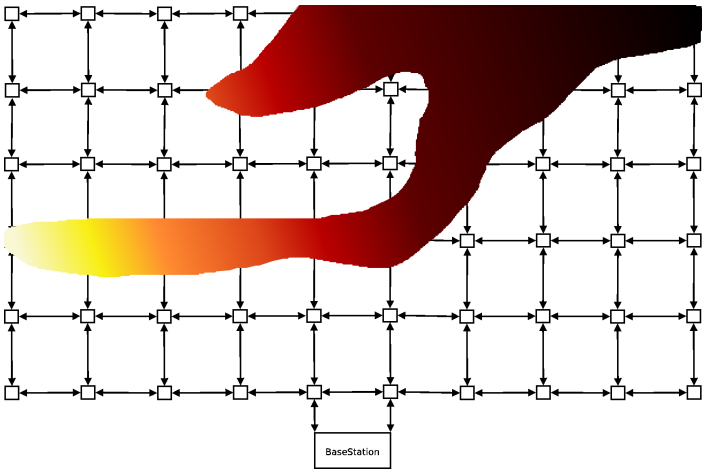


FIG. 5.11*: Déploiement simple en grille avec scénario feu*

Sur la Figure 5.12, nous représentons les résultats obtenus après analyse des données. Cette Figure représente les noeuds qui sont présents en fin de simulation, dans le tableau de résultats de la station de base.

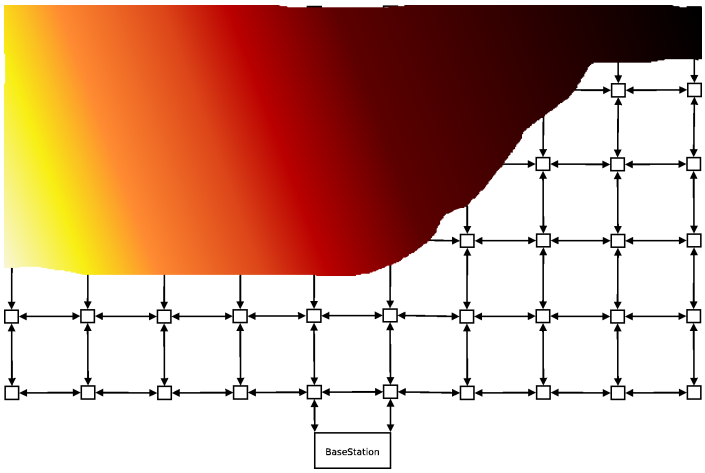


FIG. 5.12*: Résultats observés aprés analyse des données*

Nous observons sur la Figure 5.12 une zone noire, correspondant à une zone n'ayant plus transmis d'informations. Tous les noeuds ciblés comme étant défaillants ont été exclus du réseau à cause de leur incapacité à transmettre les données. Le noeud défaillant n'est plus considéré par les autres noeuds selon le principe d'éviction du protocole Xmesh (*cf* 4.3.4).

La structure simple en grille avec une seule station de base montre ses limites dans ce cas précis. En effet, le réseau ne transmet pas de réelles informations sur la localisation du feu. Les données reçues par la station de base sont contradictoires avec celles observables sur le terrain, illustrées par la Figure 5.11. Dans le cas d'un suivi du feu, la zone noire ne peut plus nous informer sur la situation réelle du feu. Cette incertitude provient de la défaillance du réseau. Les noeuds "brûlés" ne peuvent plus jouer le rôle de routeur. Les noeuds dépendants de ces derniers ne peuvent plus transmettre leurs informations jusqu'à la station de base. Le réseau est alors divisé en deux. Les données obtenues ne permettent pas de visualiser correctement la localisation du feu.

Dans l'analyse des résultats nous soulevons un autre problème. Le temps d'arrivée des messages est augmenté. En effet, nous remarquons qu'un noeud transmet ces informations toutes les 375 secondes. Ce résultat est la conséquence directe de la structure et du nombre d'entités dans le réseau. Cette valeur nous montre une autre limite du déploiement simple en grille dans l'étude feux de forêts.

Nous mettons en avant dans un cas précis la défaillance du déploiement en grille avec une simple station de base. Il semble important de réfléchir sur les stratégies de déploiement.

5.4.3 Déploiement complexe en grille

Nous proposons une nouvelle disposition des capteurs. Nous gardons une structure en grille à 60 noeuds cependant nous divisons cette structure en quatre sous-réseaux représentés par quatre couleurs : rouge, vert, bleu et gris. Chaque réseau a un nombre arbitraire de noeuds, structure que nous représentons par la Figure 5.13. Les noeuds appartenant à un réseau peuvent communiquer uniquement avec les membres de leur réseau.

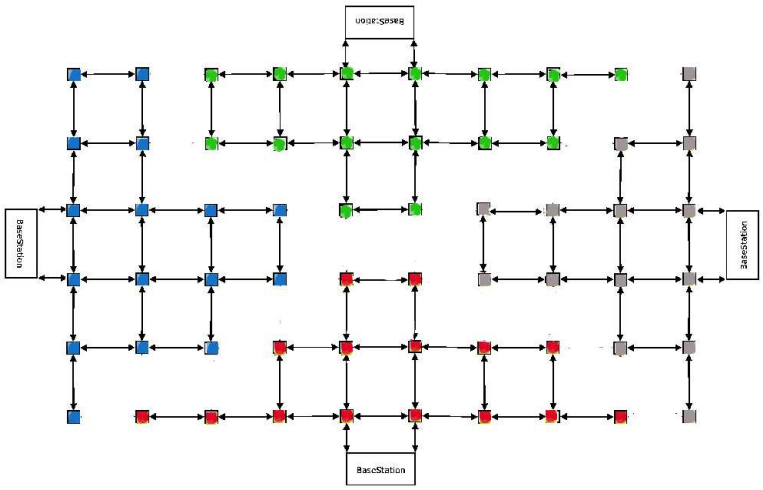


FIG. 5.13*: Déploiement complexe en grille*

Sur la Figure 5.14, nous représentons, par la zone rouge, les noeuds qui seront brûlés durant la simulation. Le scénario est identique au cas vu précédemment avec une particularité au point de vue implémentation au niveau du MA “Env”. Nous identifions quatre scénarii différents pour les quatre sous-réseaux, conséquence directe de la modification de la topologie. Les noeuds du réseau rouge auront un MA”*Env*” leur donnant un scénario de feu ; les noeuds du réseau vert auront leur scenario et ainsi de suite pour les noeuds du réseau bleu et du réseau gris.

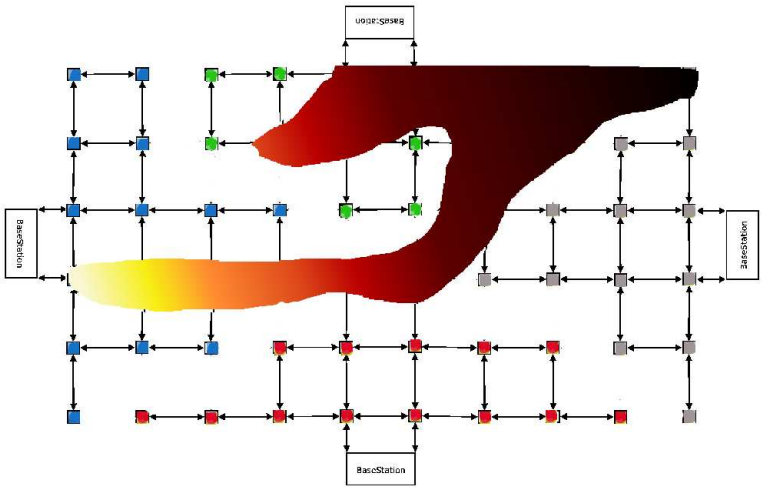


FIG. 5.14*: Déploiement complexe en grille avec scénario feu*

Les données analysées et illustrées par la Figure 5.15, montrent une réelle évolution par rapport au déploiement simple en grille. La zone noire que nous avons observé dans le déploiement simple en grille a largement diminué. La visibilité est améliorée permettant une localisation plus précise du feu. L'avantage retenu par le déploiement simple en grille est surtout la rapidité des données transmises. Au regard du déploiement à quatre sous-réseaux proposé, les données sont transmises plus rapidement sur la station de base. Le temps le plus important de transmission concerne le réseau bleu. En effet un noeud transmet son information au bout 100 secondes. Nous remarquons que ce temps est diminué de manière importante au regard du déploiement simple en grille.

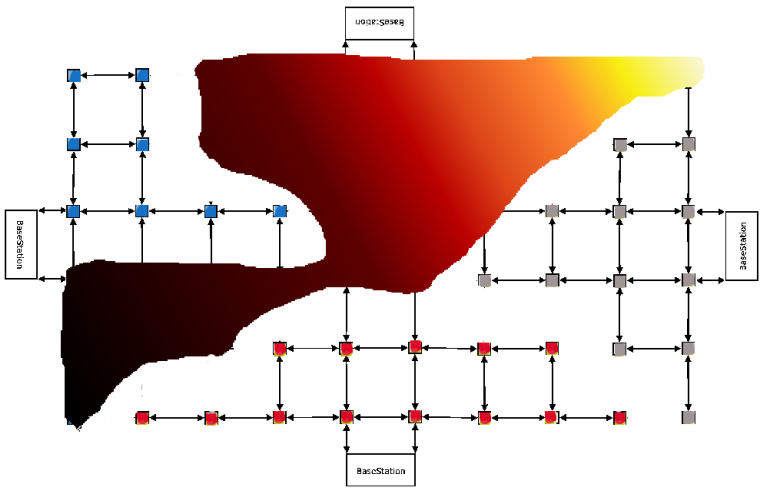


FIG. 5.15*: Résulats observés après analyse des données*

5.4.4 La particularité du déploiement simple en cercle

Nous étudions à présent le déploiement en cercle. Le déploiement que nous présentons sur la Figure 5.16 est constitué de 69 noeuds dont les connexions sont définies de manière à créer une topologie circulaire. Ce cas présente une particularité important quant aux résulats obtenus.

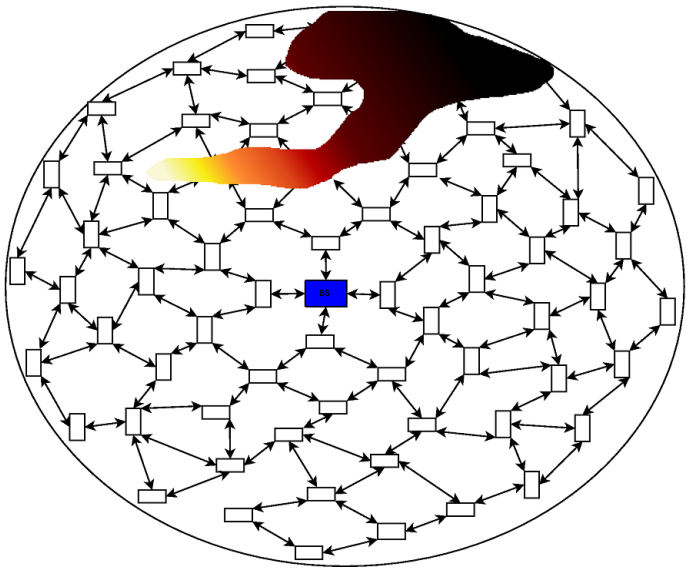


FIG. 5.16*: Déploiement simple en cercle*

Il est évident, étant donné la structure présentée (nombre d'entités importantes), que les temps d'arrivée des messages soient importants. Nous obtenons des temps supérieurs, pour une majorité de noeuds, à 540 secondes avec un maximum à 880 secondes. Ces temps sont évidemment incompatibles avec une surveillance ou une détection de feux de forêts (*cf* 1.2). Cependant la particularité apparaît sur l'analyse des résultats obtenus. En effet, la structure n'est pas mise en défaut, dans ce cas précis, sur la visualisation du feu. Nous visualisons de manière très claire la localisation du feu en fin de simulation. Les multiples voies de routage existantes autorisent les noeuds non défaillants à continuer à transmettre leurs informations à la station de base. Le temps de transmission des données et la qualité de représentation créent un paradoxe. Dans une logique

de détection et de suivi d'un feu en temps réel est-il viable d'utiliser une telle structure ? Sachant que nous désirons disposer d'informations dans un intervalle de 9 à 10 minutes, cette solution n'est pas envisageable. Ce déploiement circulaire, bien qu'efficace dans le routage des informations vers la sation de base, impose des conditions temporelles trop importantes. Pour réduire ce phénomène temporel, nous allons appliquer le principe précédent de division en sous-réseaux.

5.4.5 Déploiement complexe en cercle

Nous proposons une autre forme de déploiement : une topologie circulaire définie par 4 sous-réseaux, sachant que le maximum est de 19 entités pour un sous-réseau comme illustrée par la Figure 5.17. Nous proposons le même type de scenario que précédemment.

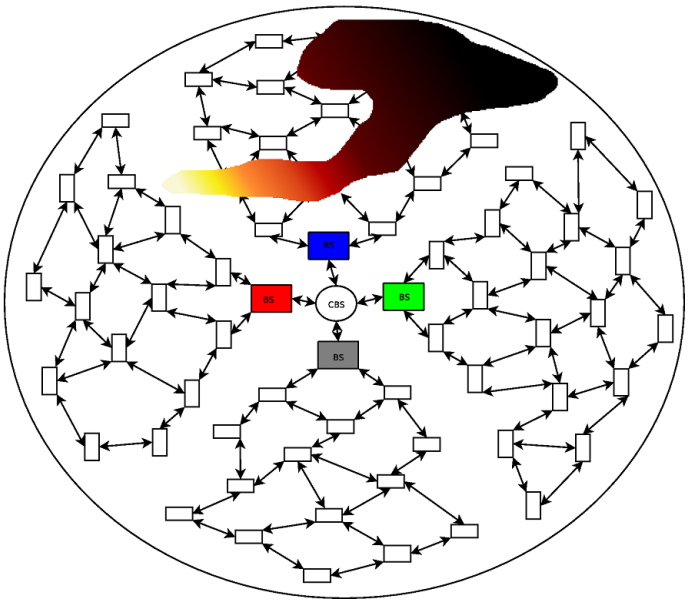


FIG. 5.17*: Déploiement complexe en cercle*

Les résultats après analyse sont présentés sur la Figure 5.18. La première analyse concerne naturellement le temps d'apparition des messages. Nous observons naturellement une nette réduction de ces temps. La diminution des vitesses de transmission des messages est la conséquence directe de la réduction du nombre d'entités dans chaque sous-réseau. Concernant la perte d'information, nous pouvons observer une perte de données sur le réseau bleu. Néanmoins, nous arrivons

à localiser de manière plus précise la position du feu par rapport au déploiement complexe en grille, même si une zone floue persiste.

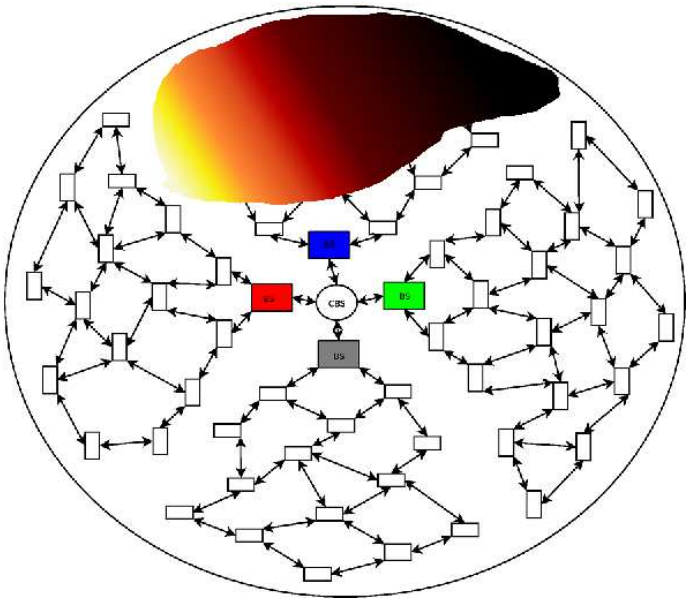


FIG. 5.18*: Résultats observés après analyse des données*

5.4.6 Discussion

Les différents résultats obtenus ont permis de soulever des points importants. En effet, nous avons vu que le feu peut avoir un effet important sur les communications du réseau, conduisant à une perte sérieuse de données. Nous admettons que les structures sont triviales mais elles fournissent cependant deux principes importants dans notre cadre d'étude :

1. la multiplication des noeuds augmente considérablement les temps d'arrivée des messages sur la station de base, ce qui pose un problème avec le concept de données en temps réel ;
2. la stratégie de déploiement va déterminer la qualité des informations transmises.

Nous présentons une classification des quatre stratégies de déploiement étudiées en fonction des caractéristiques de temps d'arrivée des messages, de la qualité de l'information (visualisation plus ou moins précise du phénomène) et de la complexité de mise en place du réseau. Nous considérons que les temps d'arrivée des messages sur la station de base (Alerte) priment sur la représentation précise du feu (suivi). La classification est présentée sur la Figure 5.19.

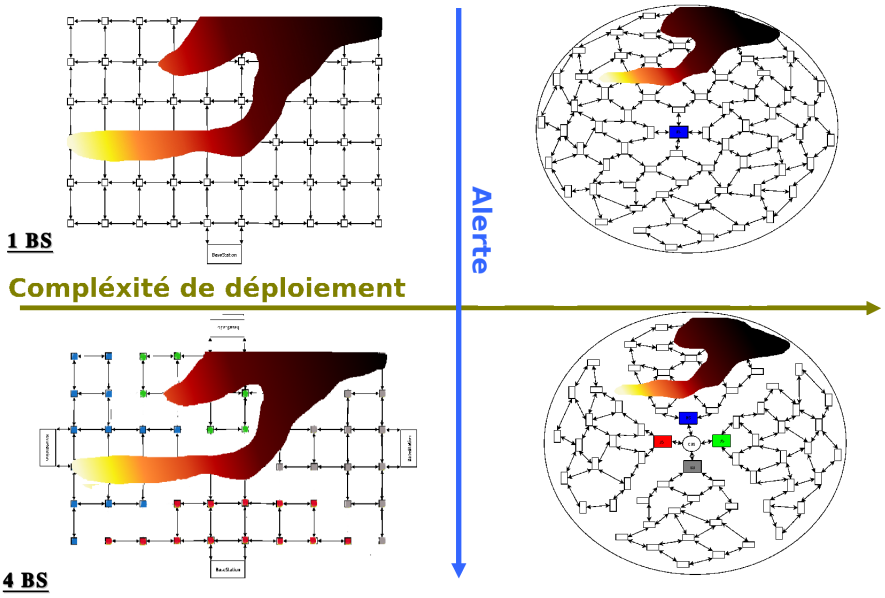


FIG. 5.19*: Classification des stratégies de déploiement*

L'objectif était d'étudier des stratégies de déploiement de réseaux de capteurs dans la problématique des feux de forêt. Nous avons déterminé les failles susceptibles de créer des dysfonctionnements du réseau. Cependant nous devons apporter des précisions sur les temps obtenus. Le processus de partage du média est défini par le protocole MAC. Nous avons implémenté notre système de réseau selon le protocole S-MAC où les temps résultants semblent trop importants. Les temps fournis par nos simulations doivent être considérés vis à vis du protocole MAC. Les temps importants obtenus lors des transmissions d'informations nécessitent sûrement de réfléchir sur la capacité du protocole S-MAC à réduire ces temps et sûrement de s'orienter vers la recherche d'un protocole MAC plus adapté. Les stratégies de déploiement proposées fournissent seulement des pistes de réflexion cependant nous pouvons affirmer qu'un déploiement de plusieurs milliers de capteurs avec une seule station de base n'est pas viable dans le suivi d'un incendie.

5.5 Conclusion

Nous avons présenté dans ce chapitre les différents résultats de simulation. Nous avons procédé premièrement à une comparaison des protocoles de routage GBR, Xmesh et VOX selon des paramètres énergétiques et selon l'activité du processeur. L'algorithme VOX, que nous avons proposé, est la technique de routage la plus efficace. Ce protocole augmente la durée de vie du réseau en favorisant un partage équitable des tâches entre les noeuds. Ses capacités et son appartenance à la famille des protocoles de routage "plat" font de l'algorithme VOX le protocole le plus approprié dans la problématique des feux de forêt. Nous avons proposé une étude théorique sur des stratégies déterministes de déploiement des réseaux de capteurs sans fil. Cette étude nous a apporté la certitude que la position et le nombre des noeuds dans un réseau étaient des facteurs essentiels dans la détection et le suivi d'un feu de forêt. Nous avons également souligné le fait que la destruction de plusieurs capteurs pouvaient entrainer une défaillance du réseau et empêcher le suivi d'un feu de forêt. Nous proposons, dans le chapitre suivant, une expérience avec un réseau de capteurs sans fil dans le cadre d'un feu réel. Cette expérience aura pour but de proposer une structure de protection des capteurs et d'évaluer les capacités du réseau dans la détection et le suivi d'un feu.

CHAPITRE 6

Tests en feu réel sur une plate-forme expérimentale

Une confrontation permanente entre théorie et expérience est une condition nécessaire à l'expression de la créativité.

Pierre Joliot

DAns les deux précédents chapitres, nous nous sommes intéressés à la modélisation et à la simulation des réseaux de capteurs sans fil. Dans ce chapitre, nous étudions le comportement d'un réseau de capteurs sans fil en présence d'un feu réel. A partir donc d'une mise en situation réelle, nous établirons pour le dispositif déployé :

- sa capacité de détection du feu ;
- sa capacité à suivre l'évolution du feu ;
- l'influence de la topologie sur les paramètres de communication ;

Nous effectuerons aussi un test de structure de protection thermique dans le but de réduire les défaillances du réseau. En effet, nous avons souligné dans le chapitre précédent, l'influence de la destruction de plusieurs capteurs sur le suivi d'un feu de forêt. Au delà de l'étude d'un réseau de capteurs sans fil en conditions réelles, nous proposons un moyen de protéger les entités du réseau. Notre dispositif est associé à un dispositif de captage de flux thermique, dispositif développé à

l'Université de Corse par l'équipe feu.

Dans une première partie, nous faisons un état de l'art sur les travaux existants en identifiant les carences de ces études. La présentation du matériel utilisé dans le cadre de cette expérience est proposée dans une deuxième partie, avec la description des composants matériels et des suites logicielles. La troisième partie présente la plate-forme expérimentale, cadre de notre expérience, avec une description précise du dispositif. Les résultats expérimentaux sont présentés et analysés dans la quatrième partie. Dans la cinquième partie, nous commentons ces résultats en soulignant les principales limites. Enfin la sixième partie conclut ce chapitre.

6.1 Travaux existants

L'utilisation de réseaux de capteurs dans la détection des feux de forêt a fait l'objet d'un certain nombre de travaux. Dans ces travaux [Doolin et Sitar, 2005, Glaser, 2004, Doolin et al., 2004, Chen et al., 2003], un système permet l'étude du suivi d'un feu. Le dispositif de base est une plate-forme MICA2 de la société Crossbow Technology. Le réseau est constitué de dix noeuds, placés chacun à environ un mètre du sol à l'extrémité d'un support. La localisation est possible grâce à un système GPS. La surface de brûlage est composée d'herbes sèches courtes sur pied. Lors de cette phase de brûlage, la destruction de deux capteurs a entrainé la défaillance d'une partie du système. Ceci nous a donc conduit à envisager l'utilisation d'une structure de protection des noeuds, structure permettant néanmoins une lecture fiable de toutes les informations. La destruction et la défaillance d'une partie du système confirme l'intérêt de notre étude théorique sur les stratégies de déploiement (*cf* 5.4). Une autre approche [Li et al., 2006] utilise le même type de dispositif mais complète l'approche précédente par une réflexion sur la communication entre la station de base et l'extérieur (par le biais d'un réseau internet). Tous ces travaux constituent une première approche et méritent d'être complétés. En effet, tester le dispositif sur le sol en le soumettant à un feu intensif, ainsi que réduire les risques de défaillance nous ont paru indispensable.

6.2 Présentation du matériel

Notre choix s'est porté sur les produits Crossbow Technology et la plate-forme MICA2. Fondé à San José en 1995, Crossbow Technology Inc. est l'un des principaux fournisseurs de solutions dans les réseaux de capteurs sans fil et systèmes de sonde à inertie. La société Crossbow offre des solutions pour relier le monde physique au monde numérique à travers les réseaux capteurs sans fil. Nous disposons de 4 capteurs Crossbow de la famille MICA2 avec une carte"capteurs"[1] de la classe MTS 420.

Nous précisons que cette étude ne doit être en aucun cas considérée comme une recommandation pour les produits de la société Crossbow Technology.

6.2.1 Description matérielle de la plate-forme MICA2

La plate forme MICA 2 est la troisième génération de capteurs de la société Crossbow Technology. Destinée spécifiquement au système de réseau sans fil, elle constitue une base de communication. Elle possède un microprocesseur, une radio, une mémoire propre et des connecteurs 51-pin comme le montre la Figure 6.1.

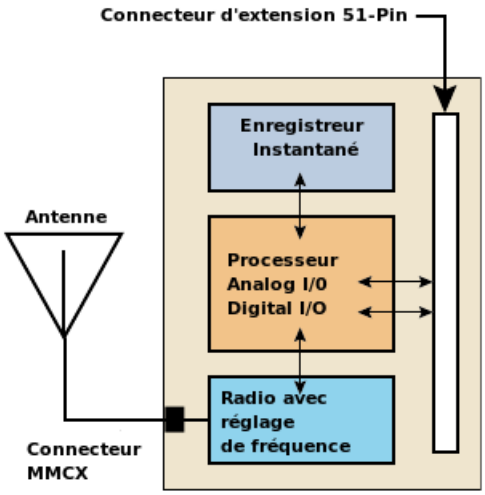


FIG. 6.1*: Bloc-Diagramme de la plate-forme MICA2*

[1]La carte "capteurs fait référence au terme anglophone *SensorBoard*, dont la traduction littérale est panneau de sonde.

Cette plate-forme représentée sur la Figure 6.2 possède un microprocesseur Atmel^R ATMega128L consommant peu d'énergie et disposant d'une mémoire Flash de 128 kB et de 4kB de SRAM. La radio est une Chipcon CC100 autorisant une transmission supérieure à 38,4 kbps. La fréquence de la transmission, utilisée dans le cadre de notre étude est 433 MHz. Elle permet des communications entre noeuds distants de 100 m à 300 mètres (champ libre). L'alimentation électrique est réalisée par deux piles AA.

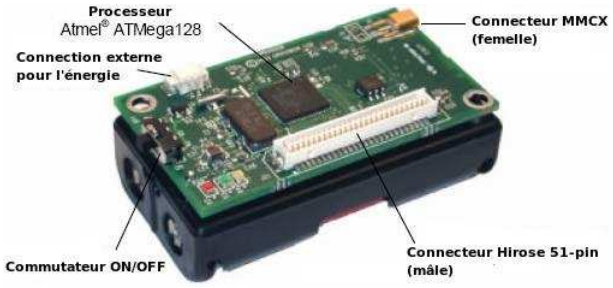


FIG. 6.2*: Plate-forme MICA2*

La plate-forme MICA2, destinée à l'analyse et la communication, doit être couplée à une carte "capteurs"[2] pour les données environnementales. Le couplage entre la plate-forme MICA2/carte "capteurs" se fait par le biais du connecteur Hirose 51-pin.

6.2.2 La carte "capteurs" : MTS 420

Notre choix s'est porté sur la gamme MTS 420, illustrée par la Figure 6.3 pour deux raisons essentielles :

1. elle dispose d'une gamme importante de capteurs ;
2. elle contient un module GPS pour la localisation géographique des entités.

Cette carte destinée à la détection environnementale intègre différents capteurs :

[2]La carte "capteurs" fait référence au terme anglophone *Sensor Board*, dont la traduction littérale est panneau de sondes

– un capteur d'humidité et de température de type Sensirion SHT11, avec un degré de précision de 3,5% pour l'humidité relative et de 0,5 ˚C pour la température ;
– un capteur de pression atmosphérique de type Intersema MS5534A, allant de 300 à 1100 mbar avec une précison de 3% ;
– un accéléromètre sur 2 axes de type ADI ADXL202 de résolution de + ou - 2 mg ;
– un capteur de lumière de type TA0S TSL2250 ;
– un module GPS de type LeadTek 9546 avec une précision de 7 mètres.

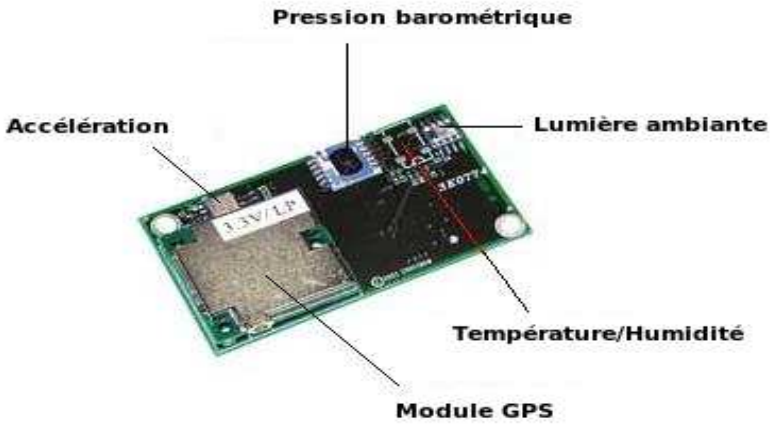


FIG. 6.3*: Panneau de sonde MTS 420*

6.2.3 La station de base MIB510

La transmission des informations du réseau à l'ordinateur se fait via la station de base (*base-station*) ou puit. Cette dernière représente l'étape finale de la transmission des données. Pour cette phase, la station de base MIB510 illustrée sur la Figure 6.4 dispose d'un port série pour la communication avec l'ordinateur. La programmation des noeuds se fait à partir de cette station qui possède un connecteur pour la plate-forme MICA2. Il est important de préciser que pour communiquer avec le réseau, la station de base doit posséder une interface de communication.

Pour cela, une plate-forme MICA2 est connectée sur la station de base par le biais du connecteur MICA2.

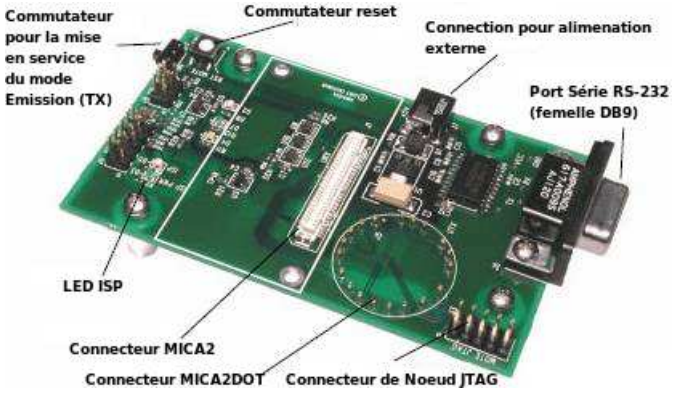


FIG. 6.4*: Station de base MIB510*

La phase de programmation d'un capteur consiste à intégrer les paramètres d'identification, de couplage avec la carte "capteurs" et de communication. Ce processus figure en Annexe 2.

6.2.4 Description de la suite logicielle

TinyOS est un système d'exploitation "Open Source" déstiné aux systèmes embarqués disposant d'un ensemble d'outils de développement de logiciel. Développé à l'origine par l'Université Californienne de Berkeley, il est maintenant considéré comme un standard pour beaucoup de réseaux de capteurs sans fil. Il respecte une architecture basée sur une association de composants, réduisant la taille du code nécessaire à sa mise en place. Cela s'inscrit dans le respect des contraintes de mémoires qu'observent les réseaux de capteurs. Pour autant, la bibliothèque de composants de TinyOS est particulièrement complète puisqu'on y retrouve des protocoles réseaux, des pilotes de capteurs et des outils d'acquisition de données. L'ensemble de ces composants peut être utilisé tel quel, et il peut aussi être adapté à une application précise. En s'appuyant sur un fonctionnement évènementiel, TinyOS propose à l'utilisateur une gestion très précise de la consommation du capteur et permet de mieux s'adapter à la nature aléatoire de la communication sans fil entre interfaces physiques. Le fonctionnement d'un système basé sur TinyOS s'appuie sur la gestion

des événements se produisant. Ainsi, l'activation de tâches, leur interruption ou encore la mise en veille du capteur s'effectuent à l'apparition d'événements. Ce fonctionnement événementiel (event-driven) s'oppose au fonctionnement dit temporel (time-driven) où les actions du système sont gérées par une horloge.

Le caractère préemptif d'un système d'exploitation précise si celui-ci permet l'interruption d'une tâche en cours. TinyOS ne gère pas ce mécanisme de préemption entre les tâches mais donne la priorité aux interruptions matérielles. Ainsi, les tâches ne s'interrompent pas entre elles, mais une interruption peut stopper l'exécution d'une tâche.

Lorsqu'un système est soumis à des contraintes temporelles strictes, celui-ci gère des niveaux de priorité dans ses tâches permettant de respecter des échéances données par son environnement. Dans le cas d'un système strict, aucune échéance ne tolère de dépassement contrairement à un système temps réel mou. TinyOS se situe au-delà de ce second type car il n'est pas prévu pour avoir un fonctionnement temps réel.

TinyOS a été conçu pour réduire au maximum la consommation en énergie du capteur. Ainsi, lorsqu'aucune tâche n'est active, il se met automatiquement en veille.

Le système d'exploitation TinyOS s'appuie sur le langage NesC [Gay et al., 2003], une variante orientée composant du langage C. Il propose une architecture basée sur des composants, permettant de réduire considérablement la taille mémoire du système et de ses applications. Chaque composant correspond à un élément matériel (LEDs, timer, ADC) et peut être réutilisé dans différentes applications. Ces applications sont des ensembles de composants associés dans un but précis. Les composants peuvent être des concepts abstraits ou bien des interfaces logicielles aux entrées-sorties matérielles de la cible étudiée (carte ou dispositif électronique). L'implémentation de composants s'effectue en déclarant des tâches, des commandes ou des événements :

– les tâches sont utilisées pour effectuer la plupart des blocs d'instruction d'une application. A l'appel d'une tâche, celle-ci va prendre place dans une file d'attente de type FIFO (First In First Out) pour y être exécutée. Comme nous l'avons vu, il n'y a pas de mécanisme de préemption entre les tâches, et une tache activée s'exécute en entier. Ce mode de fonctionnement permet de bannir les opérations pouvant bloquer le système. Par ailleurs, lorsque la file

d'attente des tâches est vide, le système d'exploitation met en veille le dispositif jusqu'au lancement de la prochaine interruption (on retrouve le fonctionnement événementiel).

– les événements sont prioritaires par rapport aux tâches et peuvent interrompre la tâche en cours d'exécution. Ils permettent de faire le lien entre les interruptions matérielles (pression d'un bouton, changement d'état d'une entrée) et les couches logicielles que constituent les tâches.

Dans la pratique, NesC permet de déclarer 2 types de composants :

– les modules qui constituent les briques élémentaires de code et implémentent une ou plusieurs interfaces. Une application peut faire appel à des fichiers de configuration pour regrouper les fonctionnalités des modules. Un fichier "*top-level configuration*" permet de faire le lien entre tous les composants ;

– les interfaces que sont des fichiers décrivant les commandes et événements proposés par le composant qui les implémente. L'utilisation des mots clefs « Use » et « Provide » au début d'un composant permet de savoir respectivement si celui-ci fait appel à une fonction de l'interface ou redéfinit son code.

Il existe de nombreuses cibles possibles pour TinyOS. Malgré leurs différences, elles respectent toutes globalement la même architecture basée sur un noyau central autour duquel s'articulent les différentes interfaces d'entrée-sortie, de communication et d'alimentation illustrées sur la Figure 6.5.

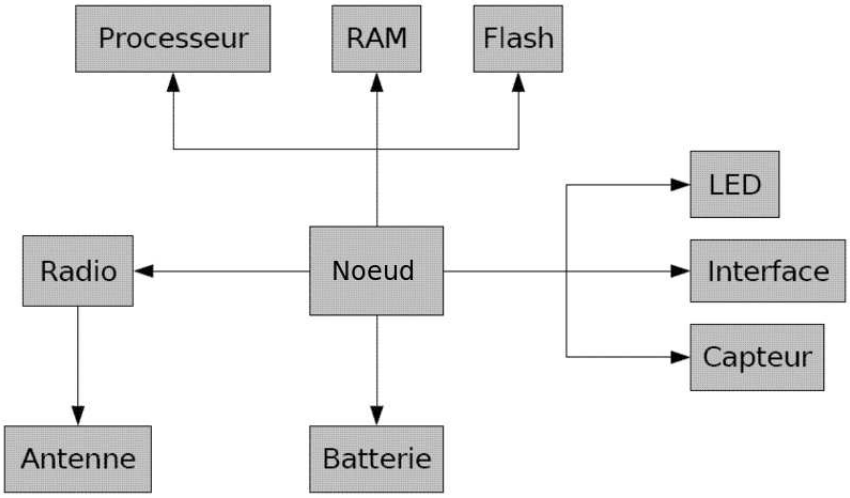


FIG. 6.5*: Architecture générale des cibles utilisant TinyOS*

L'architecture se décompose de la manière suivante :

- **Noeud, processeur, RAM et Flash** : on appelle généralement Noeud (ou mote) la carte physique utilisant TinyOS pour fonctionner. Celle-ci a pour coeur le bloc constitué du processeur et des mémoires RAM et Flash. Cet ensemble est à la base du calcul binaire et du stockage, à la fois temporaire pour les données et définitif pour le système TinyOS ;
- **Radio et antenne** : TinyOS est prévu pour mettre en place des réseaux sans fil, les équipements étudiés sont donc généralement équipés d'une radio ainsi que d'une antenne afin de se connecter à la couche physique que constitue les émissions hertziennes ;
- **DEL (*LED*), interface, capteur** : TinyOS est prévu pour mettre en place des réseaux de capteurs, on retrouve donc des équipement dotés de différents types de détecteurs et autres entrées ;
- **Batterie** : Comme tout dispositif embarqué, ceux utilisant TinyOS sont pourvus d'une alimentation autonome telle qu'une batterie.

La société Crossbox technology fournit une suite logicielle pour l'utilisation des réseaux de capteurs. Elle se décompose en deux parties :

1. le logiciel *Moteview*^{R} permettant la programmation du réseau de capteurs et l'analyse des données reçues à partir de la station de base MIB510 dans notre cas ;

2. l'éditeur de texte Crimsom Studio^R permettant l'accès au code source en NesC du systéme d'exploitation TinyOS et aux diverses applications, autorisant la redéfinition des programmes destinés au réseau de capteurs.

Ces deux logiciels vont permettre la mise en place du système.

6.3 Plate-forme expérimentale

Dans cette partie nous décrivons les expériences réalisées dans le cadre de la plate-forme expérimentale "feux de forêt". Cette étude expérimentale a été effectuée dans des conditions réelles avec un réseau de quatre capteurs. Nous testons un moyen de protection des capteurs placés dans des conditions de feu de maquis constitué de cystes et d'arbousiers coupés, végétation caractéristique de la zone méditérranéenne. Nous commençons par faire un rappel bref sur le processus de combustion. Nous décrivons ensuite la mise en place du réseau. Nous poursuivons en proposant les protections individuelles, systèmes conçus au sein de l'équipe feu. Ensuite nous présentons le protocole expérimental en détaillant le dispositif utilisé lors de cette étude.

6.3.1 Processus de combustion

Afin d'interpréter les résultats obtenus au niveau des processus de combustion, nous commençons par une description du processus de combustion intervenant lors d'un feu constitué de plusieurs étapes :

1. le matériau inerte est chauffé par transfert de chaleur, sa température augmente ;
2. le matériau se déshydrate, il y a perte de masse sans régression de surface ;
3. le matériau, soumis un flux de chaleur important, se dégrade. Lors de ce processus de dégradation chimique dit pyrolyse, il y a une libération de gaz combustibles avec perte de masse sans régression de surface ;
4. Lorsque les gaz produits par pyrolyse, atteignent la température d'ignition, une flamme se produit et la particule passe de l'état pyrolysant à l'état de combustion. Il y a régression de surface ;

5. les résidus charboneux se consumment, il y a régression de surface ;
6. les cendres restent.

Dans notre problématique de détection des feux de forêts, les deux premiers phénomènes, c'est à dire chauffage et déshydratation doivent être obligatoirement observables par le réseau de capteurs pour valider le système de détection des feux de forêt.

6.3.2 Mise en place du réseau de capteurs

Nous disposons de cinq plate-formes MICA2, de quatre cartes "capteurs" MTS 420 et d'une station de base MIB510.

Lors de la phase de mise en place de sérieux problèmes de communication sont apparus : un des capteurs a été incapable de transmettre des informations. Nous avons donc été contraints de réduire notre système à trois noeuds. Dans notre étude, ces trois noeuds ont été disposés sur le terrain et un quatrième noeud a été lié à la station de base pour la communication. La mise en place des capteurs sur la zone expérimentale a été précédée par une phase de préparation des différents éléments intervenant dans le réseau.

Chaque noeud a été programmé en lui attribuant un nom, ce nom permettant de l'identifier au cours de l'expérience (*cf* Annexe 2). Ainsi sont disposés sur le terrain trois noeuds auxquels nous donnons les noms Noeud 1, Noeud 2 et Noeud 3. Le Noeud 0, dernier programmé, correspond au noeud lié à la station d'aquisition MIB510. Cette programmation est faite à l'aide du logiciel *Moteview*^{R} .

6.3.3 Protection des capteurs

Pour soumettre ces capteurs à des températures intenses, nous les avons protégés avec des "chaussettes" réalisées l'aide de fibre ZETEX[3] dans les laboratoires de l'équipe feux. Nous avons utilisé deux types de chaussettes, illustrées par la Figure 6.6 :

– chaussette simple : chaussette en ZETEX cousue à l'aide du fil de kevlar ;

[3]ZETEX est une marque d'un textile technique haute température, haute performance, protégeant jusqu'à 600°C. Le ZETEX aluminisé est une protection contre la chaleur radiante.

– chaussette élaborée : chaussette simple + chaussette interne en ZETEX aluminisé cousue à l'aide de fil kevlar bourrée de laine céramique (matériau calorifuge utilisé pour augmenter la résistance thermique de l'enveloppe réduisant le transfert de chaleur d'un milieu à un autre).

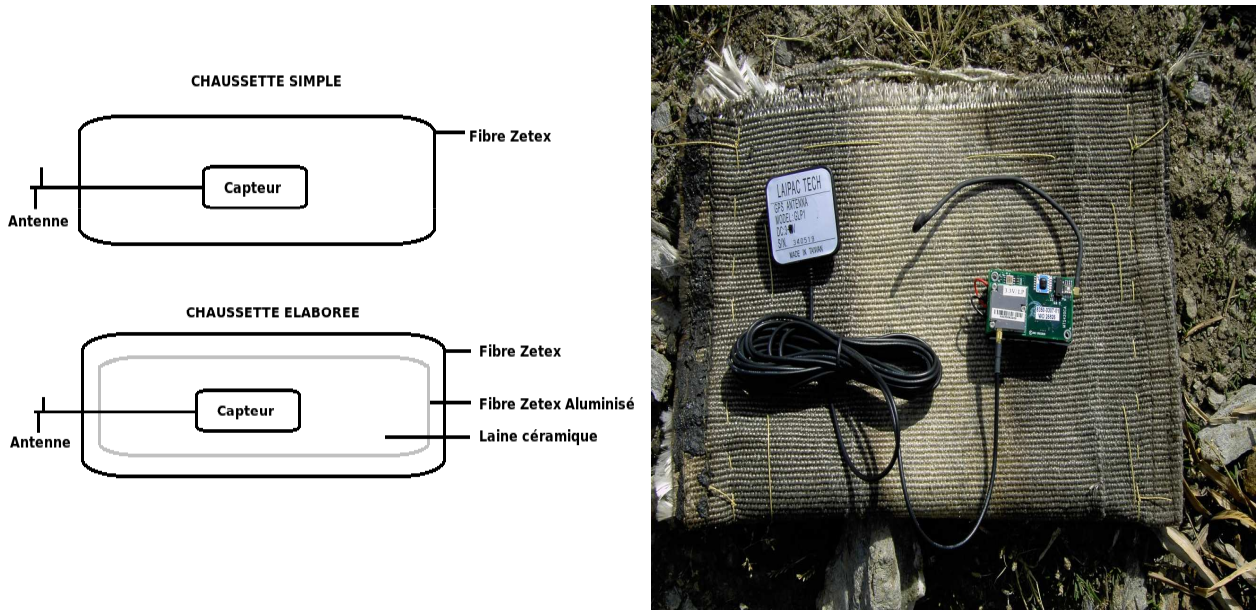


FIG. 6.6*: Structure des chaussettes*

Ces deux types de chaussettes de protection pour les capteurs nous ont permis de tester leur capacité de résistance à des conditions extrêmes de feu.

6.3.4 Protocole Expérimental

Le site expérimental est situé sur la commune de Noceta, Haute-Corse, à environ 20 km de Corté.

Nous avons placé sur le site expériemental les quatre capteurs dont nous disposions. Comme nous l'avons déjà souligné, seuls trois capteurs ont été placés sur le terrain, le dernier a été fixé sur la station de base pour recevoir les communications radio des autres noeuds. La Figure 6.7 montre le dispotif expérimental.

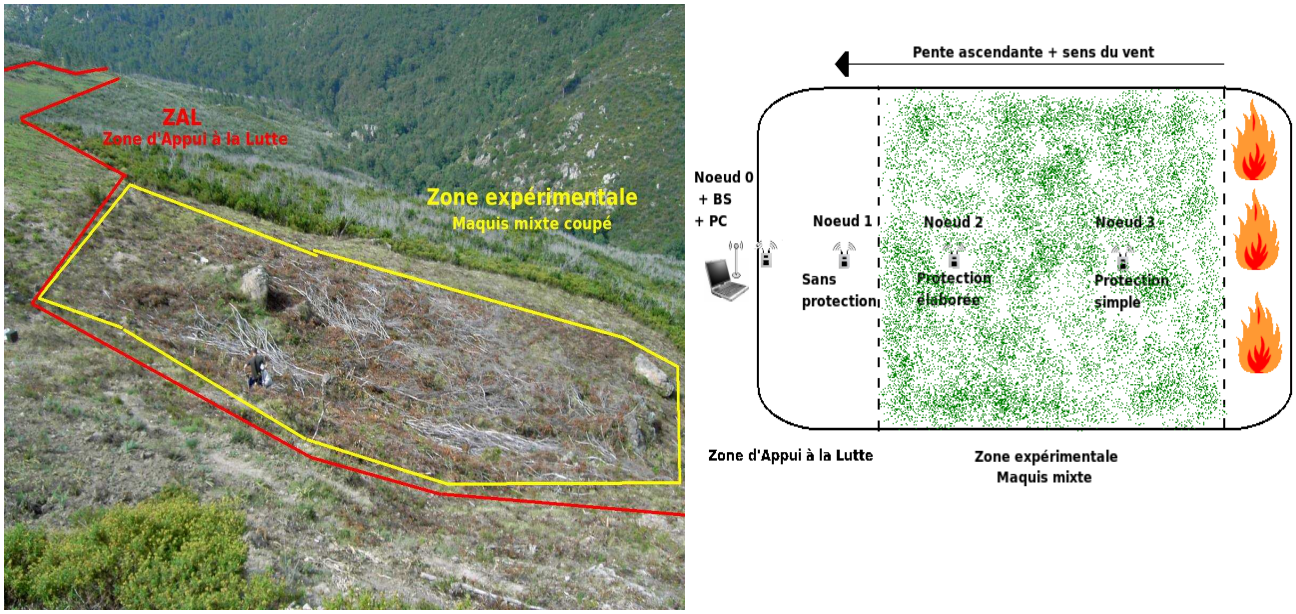


FIG. 6.7*: Zone expérimentale sur la commune de Noceta*

La zone expérimentale est divisée en deux parties :

- la Zone d'Appui à la Lutte (ZAL) : Les ZAL (anciennement appelés pare-feu) ont pour objectif de fournir un site de lutte et une zone de coupure contre les grands feux. Ces ouvrages sont formés d'un ensemble indissociable constitué d'un espace débroussaillé de 100 mètres de large et d'une voie de circulation praticable par les engins des services de lutte, et de réserves d'eau (citernes).
- la zone expérimentale : cette parcelle est recouverte de végétaux coupés (cystes, bruyères, maquis) d'une surface d'environ, 200 m^2. En pente ascendante à partir du point de mise à feux, elle bénéficie d'un vent orienté dans l'axe de la pente.

Nos capteurs ont été disposés de la manière suivante :

- l'ordinateur portable et la station de base couplée au Noeud 0 sont disposés dans la ZAL pour être à l'abri de tous risques ; l'ensemble a été protégé à l'aide d'une housse et couvert de cystes verts pour éviter tout embrasement sous l'effet de projection ;
- le Noeud 1, noeud témoin est proche de la station de base éloignée seulement de 5 mètres ; ce noeud ne possédant aucune protection a un double rôle : celui de témoin pour la mesure

des températures et celui de relais pour les autres noeuds placés plus bas dans le dispositif, pour éviter tous risques de perte de connexion ;

– le Noeud 2 avec chausette élaborée est placé à une distance de 9,80 mètres de la station de base ; son but est de détecter le passage de la flamme ;
– le Noeud 3 avec une protection basique devra lui aussi fournir des informations sur le passage de la flamme.

Les Noeuds 2 et 3 sont espacés de 7 mètres et placés dans l'axe de propagation du feu et dans le sens du vent. Ce dispositif va nous permettre d'évaluer la vitesse de propagation du feu.

6.4 Résultats

Dans cette partie nous présentons et analysons les différentes données fournies par les capteurs entre la mise à feu de la zone expérimentale (14h 08mn) et la fin de la collecte de mesures (15h 08mn). Rappelons que le Noeud 1 témoin est situé dans la ZAL, et par conséquent non soumis au feu. Le Noeud 2 avec protection élaborée et le Noeud 3 avec protection simple sont placés au coeur du feu expérimental.

6.4.1 Résultats du Noeud 1 témoin

Le Noeud 1 disposé dans la ZAL, c'est à dire au bord de la surface d'expérimentation, va servir de référence pour les températures. Les données collectées sont envoyées à la station de base puis transmises à l'ordinateur portable. Elles sont ensuite extraites et analysées. La Figure 6.8 montre les données collectées lors de l'expérience. Ces données sont la température en degré Celsius et le pourcentage d'humidité.

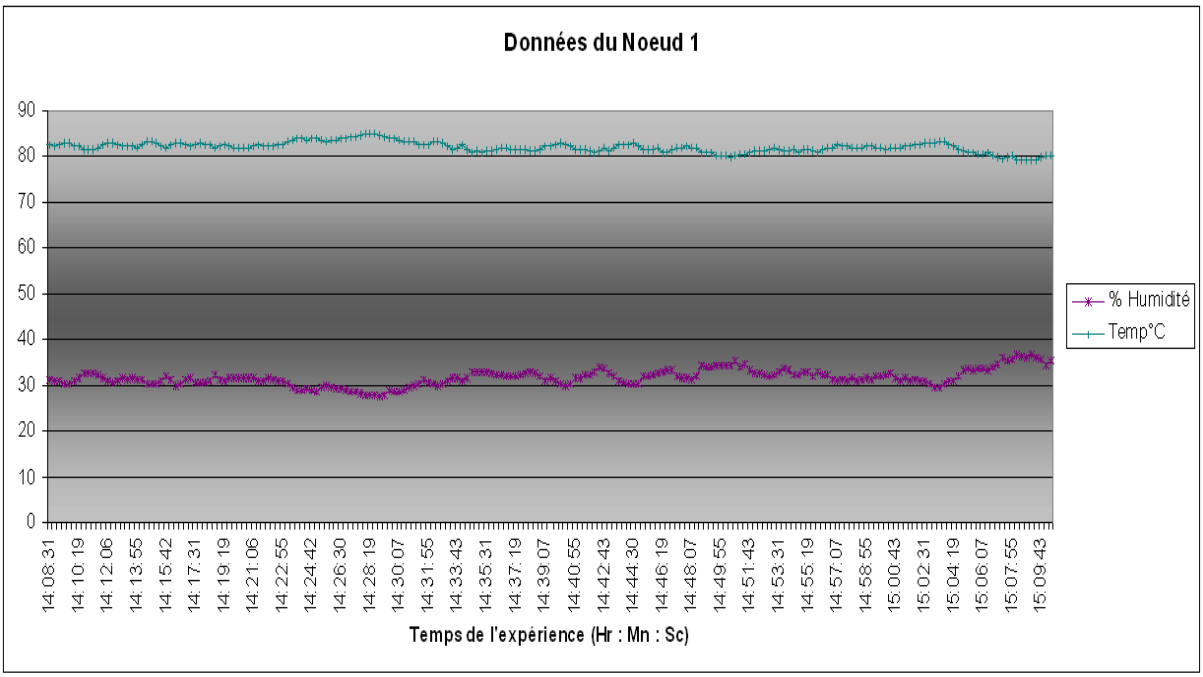


FIG. 6.8*: Résultats du Noeud 1*

Nous voyons que la température et le taux d'humidité sont relativement uniformes. Notons que ces variations sensibles sont peu significatives pour exprimer un phénomène.

La courbe de température oscille autour de 80/85˚C. Cette valeur est trop importante et elle ne reflète pas la température réel. Pour cette raison, nous n'utilisons que l'aspect de la courbe pour étudier l'apparition d'un feu. Le taux d'humidité varie entre 30 et 35 % .

Le Noeud 1 étant la référence va nous servir à visualiser le passage du feux sur les autres capteurs. Dans ce but, nous ne tiendrons compte seulement que de l'aspect de la courbe et non des valeurs.

6.4.2 Résultats du Noeud 2 avec protection élaborée

Le Noeud 2 possède une protection plus importante constituée d'une double couche de ZETEX et de ZETEX aluminisé, plus de la laine céramique. Le Noeud 2 est placé à 9m 80 du Noeud 1.

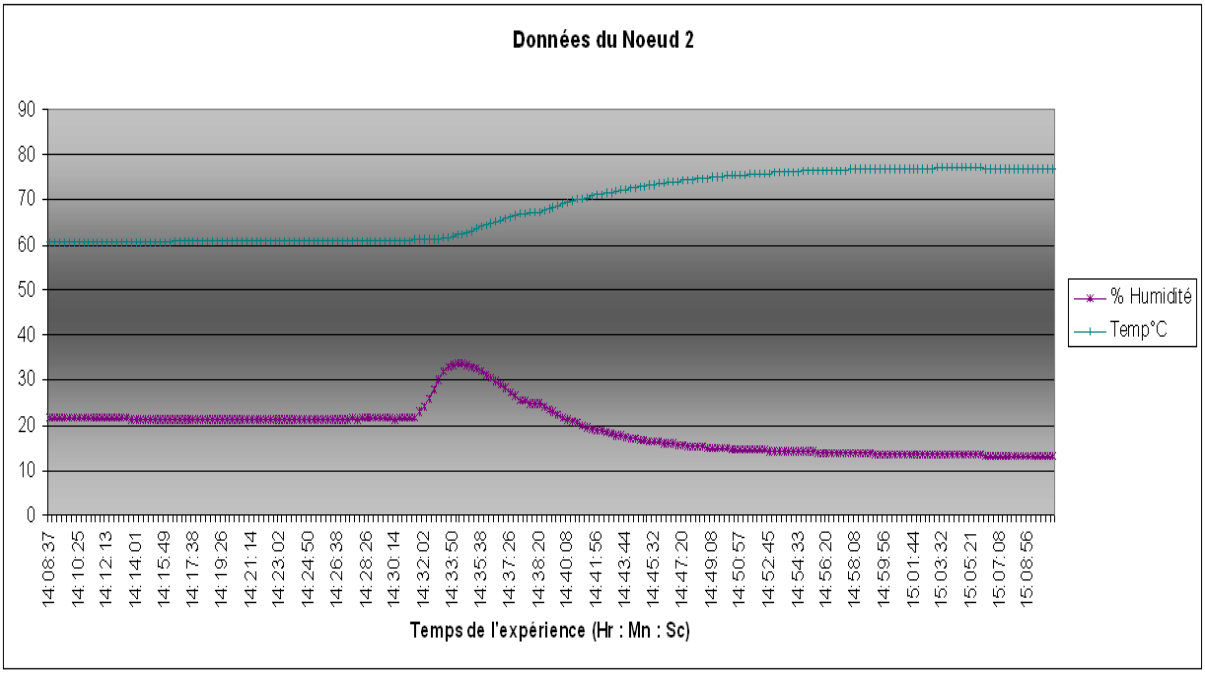


FIG. 6.9*: Résultats du Noeud 2*

La Figure 6.9 montre les résultats obtenus durant l'expérimentation. On peut d'abord noter la qualité des courbes obtenues. La courbe de température et celle du taux d'humidté sont bien coorélées. Ensuite la comparaison avec les données reçues par le Noeud 1 fait apparaître une diminution des valeurs aussi bien pour la température que pour le taux d'humidité : la température passe de 80˚C à 61˚C et le taux d'humidité passe de 30/35% à 22%. Ces différences sont sans doute des conséquences directes de la protection en Zetex.

A l'arrivée de la flamme sur le Noeud 2, on observe deux changements importants sur les deux courbes :

- la courbe de température s'élève exactement à l'arrivée du feu sur le capteur ; elle croît progressivement pour atteindre un seuil d'environ 78˚C, soit une augmentation de plus de 17˚C ; ce seuil est maintenu jusqu'à la fin de l'expérience ; on n'observe pas de chute immédiate de la température ; celle-ci ne commence qu'en fin d'expérience ; ceci est peut-être dû à un échauffement du matériel ;
- la courbe du taux d'humidité est particulièrement significative. ; elle présente un pic impor-

tant à l'arrivée du feu ; de 22%, le taux d'humidité atteint un seuil de 34% en environ deux minutes, puis redescend jusqu'à dépasser le minima de 22% et atteindre les 14% d'humidité ; cette augmentation du taux d'humidité est due au phénomène de deshydratation (phase 2 du processus de combustion).

6.4.3 Résultats du Noeud 3 avec protection simple

Le Noeud 3 possède une protection simple avec une chaussette de ZETEX. Le Noeud 3 est placé à 16m80 du Noeud 1.

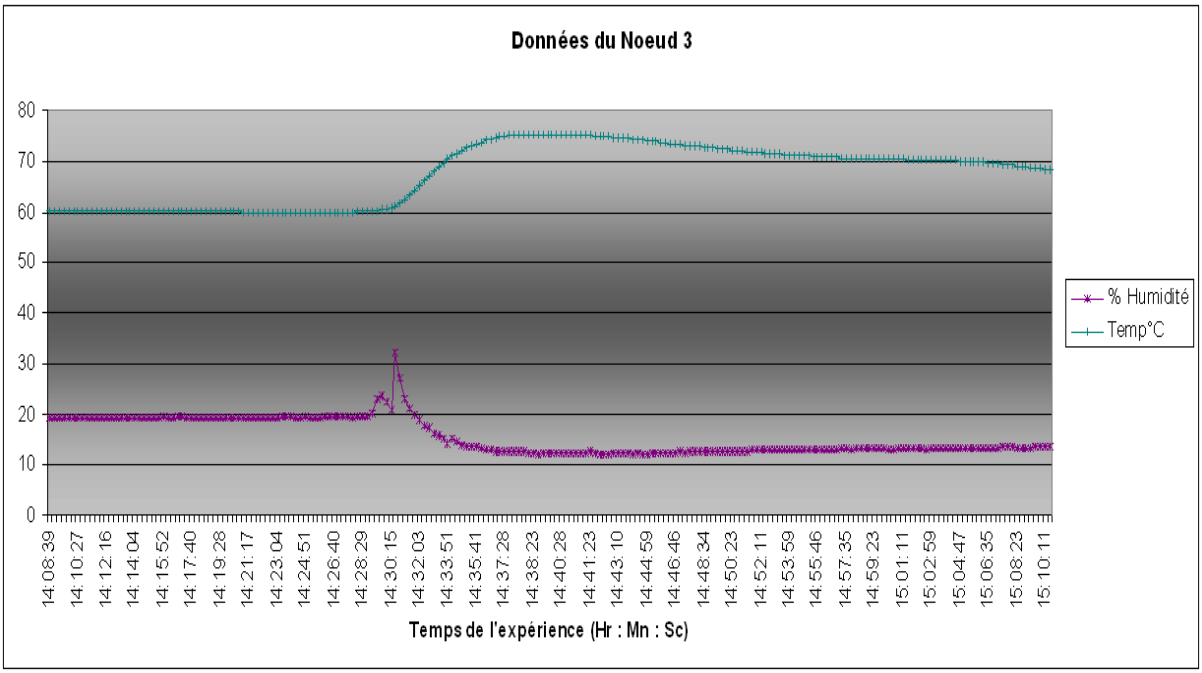


FIG. 6.10*: Résultats du Noeud 3*

La Figure 6.10 montre les courbes de température et de taux d'humidité relatives aux données collectées par le Noeud 3 durant l'heure d'expérimentation. Comme dans le cas du Noeud 2, on peut noter une cohérence des données. En effet, les courbes de température et d'humidité sont nettes et présentent des similitudes avec celles du Noeud 2.

La courbe de température présente le même seuil que celle du Noeud 3 au début de l'expérimentation, seuil proche de 60˚C. Cette courbe montre un accroîssement de la température

signalant l'arrivée de la flamme sur le capteur. La température augmente progressivement jusqu'à atteindre une température proche de 75°C. Ce seuil va se maintenir quelques minutes puis redescendre progressivement jusqu'à atteindre une valeur de 68°C en fin d'expérience.

La courbe du taux d'humidité présente le même seuil que celle du Noeud 2. A l'arrivée de la flamme sur le capteur, le taux d'humidité va croître et atteindre un premier pic à 25% puis un second à 33%. Ce taux d'humidité va ensuite décroître rapidement pour atteindre une valeur de 12%. Comme dans le cas du Noeud 2, le taux d'humidité se stabilise à une valeur inférieure à celle du départ, après le passage du feu. Les résultats du Noeud 3 sont proches de ceux du Noeud 2 et correspondent au phénomène d'évaporation de l'eau des végétaux dans le phénomène des feux.

6.4.4 Discussion

Dans le tableau qui suit et à partir des courbes relatives aux noeuds 2 et 3, nous donnons les variations de température et de taux d'humidité transmis par les capteurs 2 et 3, ces deux capteurs étant placés dans la zone de brûlage.

	Noeud 2	Noeud 3
Accroissement de la température $\Delta T(^{\circ}C.min^{-1})$	0,36	1,2
Diminution de la température$\Delta T(^{\circ}C.min^{-1})$	0	0,14
Accroissement de la température $\Delta R_H(\%.min^{-1})$	4	13
Diminution de la température $\Delta R_H(\%.min^{-1})$	0,6	0,8

TAB. 6.1*: Variations des valeurs collectées au cours du temps*

Les variations de la température ΔT ainsi que celles du taux d'humidité ΔR_H sont beaucoup plus importantes dans le cas du Noeud 3 que dans celui du Noeud 2. On constate aussi que la température fournie par le Noeud 2 reste constante jusqu'à la fin de l'expérience. A l'évidence de tels écarts viennent des protections :

– la double chaussette constitue une enveloppe de résistance thermique trop importante,
– la chaussette simple a bien protégée la carte "capteurs" et sa réponse est meilleure que celle de la double chaussette ; en effet on peut observer la chute de la température avant la fin de

l'expérience ; de plus ΔT et ΔR_H du Noeud 3 ont des valeurs plus importantes que celle du Noeud 2.

D'autre part les températures recueillies pour les trois noeuds nous semblent trop importantes. Nous avançons deux hypothèses sur ces valeurs : soit un mauvais calibrage des capteurs températures , soit un échauffement du matériel durant la longue exposition au soleil, période précédant la mise à feu. Nous observons également une différence au niveau du seuil de température entre le Noeud 1 témoin non protégé et les deux noeuds protégés. Les protections thermiques sont sans doute en cause mais on ne pouvait s'en affranchir au risque de détruire les noeuds. La protection élaborée a eu une influence sur la température. En effet, nous observons que la température du Noeud 2 n'a pas diminuée après le passage du feu et il semble que la protection élaborée soit en cause. Il est probable qu'elle a emmagasiné de la chaleur, empêchant la baisse de la température en fin d'expérience. Il est nécessaire de réaliser d'autres expériences qui nous permettront de résoudre ces problèmes relatifs au seuil. Pour ces raisons, nous nous basons, par la suite sur les variations du taux d'humidité pour détecter l'arrivée du feu. Dans l'avenir, toutes ces remarques nous permettrons de faire évoluer le système.

6.4.5 Estimation de la vitesse de propagation du feu

Les noeuds transmettent leurs données de manière périodique à la station de base. Ces informations sont enregistrées avec pour chaque donnée l'heure précise de réception. A partir de là, nous pouvons déterminer la vitesse de propagation du feu. Le dispositif et les distances sont précisés sur la Figure 6.11.

FIG. 6.11*: Distances entre les noeuds du dispositif*

Le Noeud 1 est positionné à 16m 80 du Noeud 3, et placé à 9m 80 du Noeud 2. Les distances ne sont le résultat que du placement que nous pensons optimal pour les communications. La distance séparant le Noeud 2 et le Noeud 3 (tous les deux dans la zone d'expérimentation) est de 7m. Nous avons estimé la vitesse de propagation du feu. Rappelons que la ligne formée par les capteurs est dans l'axe de propagation du feu, dans le sens de la pente et dans le sens du vent. Pour cela nous utiliserons la courbe du taux d'humidité qui présente avec ses deux pics une base pour la mesure. Nous n'utiliserons pas la courbe de température car celle du Noeud 2 nous semble trop incertaine.

Le Noeud 2 a un taux d'humidité R_H , que nous considérons comme constant de 19% entre 14h 08mn 37s et 14h 31mn 42s. Après cette mesure, nous notons une augmentation rapide de ce taux. A 14h 34mn 02s, le pic de 34% est atteint.

Le Noeud 3 envoie son premier message de début d'expérience à 14h 08mn 39s et le taux d'humidité R_H vaut 19%. Cette valeur restera constante jusqu'à 14h 29mn 04s. A 14h 31mn 03s, le pic de 33% est atteint par le taux d'humidité.

Le temps séparant le maxima de chaque courbe est de 179 secondes

A partir de ces deux temps nous allons calculer la vitesse de propagation. Le feu a donc parcouru 7 mètres en 2mn 59s, soit 700cm en 179 secondes. Les positions relatives des noeuds correspondent à une optimisation des communications entre eux. Comme nous l'avons dit précédemment, à partir de l'analyse des données recueillies, nous avons choisi d'utiliser les variations du taux d'humidité R_H pour détecter l'arrivée du feu. Ainsi le maximum de ce taux est atteint sur la courbe relative au noeud 3 à $t_3 = 14h\,31mn\,03s$ alors qu'on observe ce maximum à $t_2 = 14h\,32mn\,02s$ sur la courbe relative au Noeud 2. De là on déduit la vitesse v de propagation du feu :

$$v = \frac{d_{23}}{t_2 - t_3}$$

$$\text{avec } d_{23} = 700\,cm \text{ et } t_2 - t_3 = 179\,s \rightarrow v = 3,91\,cm.s^{-1}$$

Cette mesure est confirmée par le dispositif mis en place par l'équipe feu de l'Université de Corse qui a trouvé $v = 4\,cm.s^{-1}$. Il est évident que ce résultat devra être confirmé lors d'autres

expériences mais la concordance de la valeur obtenue à partir de notre dispositif et celle de l'équipe feu nous laisse envisager de réelles capacités dans l'estimation de la vitesse de propagation du feu.

6.4.6 Routage dans le réseau déployé

Nous représentons dans cette partie la fréquence d'élection du voisin idéal durant l'expérimentation, illustrée par la Figure 6.12. Ces paramètres sont à mettre en relation avec nos résultats de simulation.

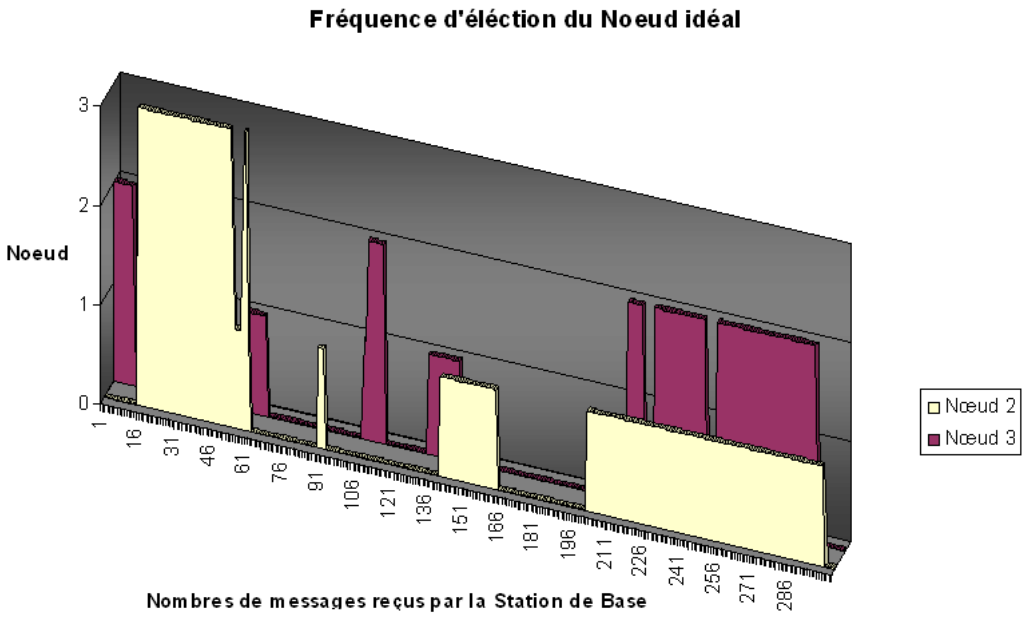


FIG. 6.12*: Fréquence d'élection du voisin idéal*

Nous observons durant l'expérimentation, que les Noeuds 2 et 3 sont dépendants de la qualité de la transmission. Nous pouvons distinguer de manière périodique que le Noeud 2, utilise le Noeud 3 pour communiquer avec la station de base. En effet, le Noeud 2 ne peut, à certains moments, ni communiquer avec le Noeud 1 ni directement avec la station de base. Ceci est la conséquence directe de l'antenne. Le moindre mouvement de cette dernière peut couper la communication avec le Noeud 1 ou la station de base.

Cependant la fréquence d'élection du voisin idéal rejoint nos résultats de simulation. Nous observons une faible fréquence de changement de voisin idéal. Le protocole Xmesh, n'utilisant que la fréquence et le nombre de sauts pour classer le voisinage d'un noeud, n'autorise par conséquent que peu de changements dans les tables de routage. Ce résultat confirme de manière claire le problème du protocole Xmesh, vis à vis du partage des tâches dans le réseau, que nous avions mis en évidence (*cf* 5.3.2). Ce résultat nécessite d'être compléter, notamment par l'étude d'un réseau avec un nombre plus important d'entités. En effet, le processus de sélection du voisinage ne pourra apparaître clairement que si le réseau est assez important tant par l'étendue de la zone d'étude que part le nombre de noeuds.

6.5 Synthèse

La mise en oeuvre du système de réseau de capteurs nous a permis de vérifier trois points importants :

1. La communication entre noeuds. Lors de nos premiers essais, nous avions constaté que le moindre accident de terrain, rocher ou simple dépression, induisait une rupture de liaison entre capteurs. La position relative des capteurs était donc un paramètre critique. Pour cela, lors du déploiement, nous avons veillé à ce que l'antenne de chaque carte capteur " voit" les autres antennes. La mise en place du dispositif doit tenir compte de la topologie du terrain car cette dernière influence grandement les liaisons entre capteurs.
2. Le système de protection. Ce système de protection a parfaitement joué son rôle. Les deux capteurs soumis à d'importantes températures ont pu résister sans réelles dégradations. La comparaison entre les données fournies par les trois noeuds émetteurs nous impose les remarques suivantes : La double chaussette est-elle bien nécessaire ? Son inertie thermique n'est-elle pas trop importante ? Nous avons, par contre, constaté des dégradations au niveau des antennes placées hors des protections dans le but d'optimiser les liaisons entre capteurs. La protection plastique de ces antennes a brûlée mais ceci a été sans conséquence sur la communication entre noeuds ainsi que sur la transmission des informations.

3. la confirmation que le protocole Xmesh ne prend pas un réel partage des tâches entre les noeuds dans le réseau. Cependant ces résultats méritent une étude à plus grande échelle.

Les courbes obtenues nous laissent entrevoir une réelle capacité d'un tel dispositif dans la détection des feux de forêts. Les courbes correspondant aux Noeuds 2 et Noeud 3 sont concluantes sur plusieurs points. Elles ont permis de déceler l'arrivée et le passage de la flamme.

Nous avons vu sur les courbes de température, malgré des valeurs trop importantes, une augmentation soudaine et rapide de la température. Cette variation de la courbe matérialise l'arrivée du feu sur les différents noeuds. Un problème concerne la courbe de température du Noeud 2 qui détecte l'arrivée de la flamme mais ne nous permet pas de déterminer la fin de son passage. La température ne régresse pas rapidement malgré la chute de la température. Bien que possédant la protection la plus élaborée, le Noeud 2 semble avoir subi un échauffement important. Les valeurs de températures sont trop importantes. Il est nécessaire de refaire des expériences pour vérifier si ces problèmes sont dus à un mauvais calibrage des capteurs ou bien dus à une défaillance de ces derniers face à un échauffement du matériel au soleil.

Les courbes exprimant le pourcentage d'humidité dans l'air ont mis en évidence le phénomène de déshydratation survenant avant l'arrivée du feu. Les courbes des Noeuds 2 et 3 ont montré une augmentation du pourcentage d'humidité avant l'augmentation de la courbe de température, correspondant à l'embrasement des matériaux. La courbe d'humidité du Noeud 3 présente un décrochage au début de la déshydration que nous ne pouvons identifier et que nous attribuerons éventuellement à une défaillance passagère du capteur. Ceci est à vérifier par de nouvelles expériences. La chute importante du pourcentage d'humidité après le passage du feu peut être attribuée à un asséchement de l'air ambiant et à une disparition de végétaux dans la zone brulée.

Globalement nous pouvons dire que le dispositif peut permettre de détecter l'arrivée du feu et d'étudier son passage (grâce aux "chaussettes") par l'analyse de la variation de la température et du pourcentage d'humidité de l'air. La brusque augmentation de ces deux données nous autorise à dire que la zone concernée est le siège d'un feu de forêt.

Ce dispositif, dans le cadre de cette étude, peut nous aider à estimer une vitesse de propagation. Ces résultats méritent d'être complétés. Nous pouvons également espérer déterminer le sens de

propagation du feu. Dans notre dispositif expérimental, les capteurs étaient placés de façon à se trouver dans le sens de propagation. Si nous appliquons une stratégie de déploiement plus importante de la zone expérimentale, les capteurs détectant le passage du feu peuvent déterminer le sens de propagation de l'incendie.

6.6 Conclusion

La partie expérimentale constitue un élément complémentaire indispensable à notre étude théorique. Les données recueillies autorisent un optimisme raisonnable quant à la future utilisation des réseaux de capteurs sans fil dans la lutte contre les feux de forêt. La qualité des informations, même si elles méritent d'être complétées, montrent que deux phénomènes physiques (déshydratation et combustion), propres au phénomène feu, sont détectés et parfaitement observables sur les courbes obtenues. Le sytème de protection pour les capteurs face au phénomène feu de forêt a montré ses capacités et représente une réelle avancée dans l'étude du phénomène. En protégeant naturellement les structures dans notre cadre expérimental, il améliore la qualité des données mesurées. Cependant cette étude n'est qu'une première étape qui doit être complétée par le test d'un réseau de capteurs sans fil plus important, qui constitue l'une de nos perspectives majeures de recherche.

Conclusion Générale

"La recherche procède par des moments distincts et durables, intuition, aveuglement, exaltation et fièvre. Elle aboutit un jour à cette joie, et connaît cette joie celui qui a vécu des moments singuliers".

Comment je vois le monde, *Albert Einstein*

L'objectif de ce travail était de valider l'utilisation des réseaux de capteurs sans fil dans le cadre de la problématique des feux de forêt. Le but avoué était de mettre en adéquation un protocole de routage pertinent et une stratégie de déploiement afin d'augmenter l'efficacité dans la détection, la prévision et le suivi d'un feu de forêt.

Pour nous permettre d'étudier un réseau de capteurs dans des conditions de feux de forêt, dans un premier temps, la simulation est apparue comme nécessaire face à une utilisation coûteuse du réseau dans des conditions réelles. Certaines carences des outils existants nous ont conduit à définir un nouvel outil. En effet, ils sont naturellement efficaces pour étudier un réseau de capteurs sans fil mais s'avèrent parfois incomplets (absence de certains composants, pas de prise en compte de l'énergie) et inutilisable (pas de possibilité d'étudier les stratégies de déploiement et de possiblité de travailler sur des conditions environnementales spécifiques) dans le cadre de notre étude. Dans un souci de généricité, de "modifiabilité", de modularité et de simplicité d'implémentation, nous avons fait le choix de créer un nouvel outil en basant notre approche sur un formalisme

unificateur dans le domaine de la modélisation et de la simulation : DEVS. A partir de ce formalisme, nous avons proposé une application orientée objet permettant la simulation de réseaux de capteurs. Dans un second temps, nous avons testé un réseau de capteurs dans des conditions réelles en évaluant les capacités de détection et de suivi d'un feu. Nos travaux se sont axés sur cinq sous-objectifs précis :

- proposer un modèle de capteur selon une description des différents composants par le biais du formalisme à événements discrets DEVS ;
- implémenter un nouvel outil : **DEVS-WSN** ;
- étudier des familles de protocoles de routage propres au domaine des réseaux de capteurs sans fil dans la problématique des feux de forêt grâce à l'outil **DEVS-WSN** ;
- proposer un algorithme de routage, VOX, basé sur une triple métrique ayant pour but l'augmentation de la durée de vie du réseau, et comparer ce dernier par aux protocoles GBR et Xmesh à l'aide d'une simulation à événements discrets ;
- analyser des stratégies de déploiement dans la problématique des feux de forêts et effectuer une classification selon les critères de complexité de mise en place et de fiabilité de l'information transmise ;
- tester en conditions réelles un réseau de capteurs MICA2 et une structure de protection sur une plate-forme expérimentale.

Bilan des travaux

Les résultats obtenus nous autorisent à penser que les objectifs ont été atteints.

Premièrement l'outil **DEVS-WSN** que nous avons développé, permet de simuler un réseau de capteurs et de proposer à la communauté du domaine, un outil modulaire et générique.

Deuxièmement, la définition de l'algorithme VOX améliore de manière certaine la durée de vie en basant son processus de discrimination sur une triple métrique. En effet, les protocoles GBR et Xmesh, utilisant respectivement une simple et une double métrique, se sont révélés peu performants dans le partage des tâches inhérentes au routage et à la survie du réseau. L'algorithme VOX se révèle plus efficace face à ces deux critères.

Troisièmement, il était nécessaire de compléter ce travail par une analyse des stratégies déterministes de déploiement face aux feux de forêt. Nous avons établi une classification de quatre stratégies de déploiement en essayant de déterminer, lesquelles pourraient permettre le suivi d'un incendie.

Quatrièmement nous avons réalisé des tests sur une plate-forme expérimentale, pour étudier le comportement réel d'un réseau de capteurs sans fil dans des conditions réelles d'un feu de forêt. Cela nous a amené à concevoir un système de protection et à identifier certaines contraintes de communication comme la topologie du terrain ou l'orientation des antennes. Les résultats obtenus ont été concluants et ont permis de compléter notre étude théorique.

Nous nous devions de répondre à trois questions majeures dans l'utilisation des réseaux de capteurs de fil dans le cas d'un incendie :

1. Est-il possible de **prévoir** l'éclosion d'un feu à l'aide d'un réseau de capteurs sans fil ?

   La réponse à cette question est ***Oui***. En effet, les réseaux de capteurs sont capables, grâce à leur carte "capteurs" de collecter des données environnementales. Un réseau déployé va transmettre les données du terrain jusqu'à un utliisateur par le biais d'une communication de proche en proche. Il peut fournir des indications sur une zone en déterminant des paramètres environnementaux favorables à l'éclosion d'un feu de forêt. Par exemple les données collectées peuvent nous informer d'une température élevée et d'un taux d'humidité faible, deux paramètres qui favorisent l'éclosion d'un feu. Ainsi un réseau de capteurs sans fil peut permettre aux pompiers de déterminer les zones sensibles et de disposer leurs moyens de lutte en conséquence. Seulement ce réseau doit avoir une durée de vie assez longue pour permettre une surveillance à long terme. La famille des protocoles de routage dits "plats", protocoles que nous avons identifiés comme les plus robustes dans le cadre d'un incendie, ne se sont pas avérés économes en énergie. La durée de vie du réseau est dépendante des ressources énergétiques de chaque noeud. La proposition du protocole VOX augmente ce temps de vie en favorisant un partage plus équitable des tâches entre les entités. Les "noeuds VOX" permettent non seulement une communication fiable (qualité de lien prise en compte et chemin le plus court) mais de plus ils économisent les ressources énergétiques. Ce proto-

cole, issu de la famille des protocoles dits “plats”, peut être considéré application-dépendant, car développé dans la problématique des feux.

2. Est-il possible de **détecter** l’éclosion d’un feu à l’aide d’un réseau de capteurs sans fil ?

   La réponse à cette question est ***Oui***. Les réseaux de capteurs sont capables de détecter un feu de forêt. Nous avons observé dans notre cadre expérimental, qu’à l’arrivée d’un feu, la température et le taux d’humidité subissaient de significatives variations représentant le phénomène feux de forêt. Ces deux paramètres peuvent donc être utilisés dans la détection des feux de forêt. Ces tests nous ont permis de valider la capacité de détection du feu. Cependant nous avons dû protéger le matériel pour qu’il ne soit pas détruit durant l’incendie. Les capteurs sans protection auraient-ils pu détecter le phénomène ? Cela mérite sûrement une réponse. Mais était-il raisonnable de brûler les capteurs durant l’expérimentation ? Nous ne le pensons pas.

3. Est-il possible de **suivre** l’évolution d’un feu à l’aide d’un réseau de capteurs sans fil ?

   Pour cette question, nous nuancerons nos propos en répondant : ***Oui mais sous certaines conditions***. Les résultats expérimentaux nous permettent de dire qu’il est possible, avec des noeuds protégés, d’étudier le passage d’un feu et d’estimer sa vitesse de propagation. Cependant, nous avons identifié plusieurs contraintes. La première concerne le protocole de routage. Nous avons caractérisé une seule famille de protocoles capable d’autoriser un suivi efficace du phénomène feu. La famille des protocoles dit “plat” offre des garanties de survie du réseau. Considérant que tous les noeuds ont le même rôle et ne basant pas leur stratégie de routage sur des données géographiques, ces protocoles éliminent un certain nombre de défaillances (destruction, panne du module de localisation). L’algorithme de routage VOX offre des garanties sur une transmission des données durant un feu. Les stratégies de déploiement se révèlent également être un paramètre important pour le suivi des feux de forêt. En effet nous avons soulevé certaines contraintes dans le cadre d’un positionnement déterministe des capteurs. L’utilisation de réseaux constitués de plusieurs noeuds avec une seule station de base ne semble pas appropriée pour le suivi d’un feu de forêt. La destruction de quelques entités peut entrainer une défaillance importante du réseau, empéchant une visua-

lisation correcte du phénomène. Dans le cadre de la prévision (aucun phénomène de feu, surveillance environnementale basique) et de la détection (possibilité de destruction d'un seul noeud au point d'éclosion de l'incendie), la visualisation du phénomène n'est pas influencée. Notre réflexion sur ce sujet nous amène à dire que l'utilisation de plusieurs petits réseaux déployés en cercle est plus appropriée qu'une structure importante en grille. Non seulement, cette disposition diminue le risque de pertes d'informations dans le cas d'un suivi, mais de plus elle permet de réduire les temps de transmission des informations pour la prévision, la détection et le suivi de l'incendie. La notion de temps est importante , car elle va autoriser une action plus ou moins rapide des secours. De ce constat, nous affirmons que le déploiement d'un réseau de capteurs sans fil est donc dépendant de l'application ou de la tâche à accomplir. Notre raisonnement sur une "application feu" des réseaux de capteurs (dans le cadre d'une destruction de capteur) peut être appliqué sur d'autres phénomènes environnementaux de même odre, tels que les avalanches ou les tsunami.

Perspectives de recherche

L'application DEVS développée est un prototype et nécessite naturellement des améliorations. En effet, tous les modèles, même s'ils se sont montrés efficaces, doivent être améliorés. Les caractéristiques de chaque composant matériel doivent être complétées pour aboutir à des résultats plus précis. Par exemple, le modèle de batterie est basé sur un modèle linéaire. Il esiste un phénomène de relaxation [Park et al., 2001], qui permet lors de la phase de non-activité du noeud de recharger la batterie. Notre modèle ne le prend pas en compte. Il serait intéressant d'étudier ce phénomène pour avoir un consommation énergétique plus précise lors de la simulation. Le protocole MAC sert aux partages du médium entre les noeuds lors de la phase de communication. Notre approche est basée sur le protocole S-MAC. Il serait intéressant d'envisager un autre protocole MAC et d'étudier son action sur les communications. Nous savons, selon [Demirkol et al., 2005], que les protocoles influencent de manière certaine la consommation énergétique. Il serait intéressant d'envisager un autre protocole MAC et d'étudier son impact sur le réseau. Notre outil ne possède pas encore d'interfaces graphiques et il est nécéssaire de combler ce manque. Dans le cadre de la

thèse de Mr Laurent Capocchi [Capocchi, 2005], une interface graphique a été développée pour la simulation concurrente de fautes comportementales. La structure en paquetages de **DEVS-WSN** peut permettre une intégration rapide dans l'interface. Il serait également judicieux de pouvoir automatiser le traitement des données qui arrivent sur la station de base. Les résultats que nous avons présentés sont le fruit d'une collecte et d'un traitement long et fastidieux de nombreuses données. Ce point nécessite un réel effort d'amélioration en particulier à la visualisation instantanée des données collectées sur la station de base. Concernant le côté applicatif des réseaux de capteurs sans fil, il est nécessaire de compléter notre étude par un test sur un nombre de capteurs plus important, autorisant l'analyse de capacités de routage et des stratégies de déploiement. L'augmentation du nombre de capteurs est prévue et nous prévoyons dans un futur très proche une expérience avec 30 capteurs. De plus, nous envisageons d'implémenter l'algorithme VOX à la plate-forme MICA2, par le biais du *SE* TinyOS. L'implémentation ne sera pas complexe. En effet, comme son nom l'indique, l'algorithme VOX (Variant Of Xmesh) est un protocole dérivé de Xmesh, technique de routage implémenté par défaut sur les plates-formes Crossbow, au sein des applications de TinyOS. Seules la création de composants "énergie", la modification de la gestion des files et la modification de l'algorithme de sélection des voisins semblent nécéssaires.

Comme le furent les chercheurs de workshop *DSN* de 1978 [DSN, 1978] il y a 30 ans, nous sommes incapables d'entrevoir les futures applications dans ce domaine. Cependant, ces futures applications, ayant pour but la surveillance environnementale, ne doivent pas être des facteurs de pollution. Premièrement, la taille actuelle des capteurs, de l'ordre du centimètre, rend les entités du réseau visibles sur une zone de surveillance ce qui crée une "pollution visuelle" importante. Deuxièmement, les capteurs actuels utilisent des piles alcalines ou des batteries Lithium pour alimenter tous les composants. Dans le cadre d'un déploiement de plusieurs entités cela peut ammener un facteur de "pollution chimique". Il faut que la composition des matériaux et le type de ressource utilisée soient pris en considération.

Les travaux de recherche pour la miniaturisation des composants, pour l'augmentation des capacités de transmission et de calcul nous amènent vers de nouveaux horizons applicatifs. Ces futures applications peuvent être entraperçues à travers le projet Smart Dust et les travaux de Brett

Warneke [Warneke et al., 2001]. Le concept initial de la poussière communicante électronique "*Smart Dust*" est crédité aux chercheurs à l'Université de la Californie Berkeley. Leur but est de réaliser un système d'un millimètre cube (taille d'un grain de sable) capable d'être déployé en réseau (millier de noeuds) et contenant les parties capteurs, communication bidirectionnelle, électronique et traitement ainsi que la récupération et le stockage de l'énergie solaire. Le tout devant être réalisé par le biais des microtechnologies, où chaque grain doit pouvoir assurer la surveillance d'une zone de 30 m^2. Il semble que, dans le domaine des réseaux de capteurs sans fil, le meilleur reste à venir.

Liste des publications

Journaux avec comité de lecture

- T. Antoine-Santoni, J.F Santucci, E. De Gentili, B. Costa “Discrete Event Modeling and Simulation of Wireless Sensor Network Performance” in Simulation : Transactions of the Society for Modeling and Simulation International(*procédure de soumission en cours*).
- T. Antoine-Santoni, Xavier Silvani, Frédéric Morandini, E. De Gentili “A Wireless Sensor Network for natural fire detection and fire spread measurements in the field” in International Journal of Fire Safety (*procédure de soumission en cours*).

Conférences Internationales avec comité de lecture

- T. Antoine-Santoni, J.F Santucci, E. De Gentili, B. Costa “Wildfire impact on deterministic deployment of Wireless Sensor Network by a discrete event simulation” in the 14th IEEE Mediterranean Electrotechnical Conference, Ajaccio, France (*procédure de soumission en cours*).
- T. Antoine-Santoni, J.F Santucci, E. De Gentili, B. Costa “DEVS-WSN : a discrete event approach for Wireless Sensor Network simulation” in the 6th ACS/IEEE International Conference on Computer Systems and Applications, 2008, Quatar (*procédure de soumission en cours*).

- T. Antoine-Santoni, J.F Santucci, E. De Gentili, B. Costa "Simulation and visualization method of Wireless Sensor Network performance" in Proceedings of International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2007) , SCS, IEEE, pp 476-483, San Diego, California, USA (***faisant partie de la sélection des meilleurs articles, considéré pour une version longue dans Transactions of the Society for Modeling and Simulation International***).
- T. Antoine-Santoni, J.F Santucci, E. De Gentili, B. Costa "Discrete event simulation of a routing protocol in Wireless Sensor Network." in Proceedings of Method and Simulation Visualization Conference 2007, part of Worlcomp 07 , pp 94-100, June 2007, Las Vegas, USA.
- T. Antoine-Santoni, F. Bernardi, F. Giamarchi "General Methodology for Metabolic Pathways Modeling and Simulation using Discrete Event Approach. Example on glycolysis of Yeast" in Proceedings of BIOCOMP 2007, part of Worlcomp 07, pp 58-64, June 2007, Las Vegas, USA.
- T. Antoine-Santoni, J.F Santucci, E. De Gentili, B. Costa "Modelling & Simulation oriented components of Wireless Sensor Network using DEVS formalism." in Proceedings of DEVS Symposium, part of SpringSim 2007 , March 2007, Norfolk, Virginia, USA.
- T. Antoine-Santoni, J.F Santucci, E. De Gentili, B. Costa "Using Wireless Sensor Network for Wildfire detection. An discrete event approach of an evironmental monitoring tool" in Proceedings of First IEEE International Symposium on Environment Identities and Mediteranean area (ISEIM 2006) , July 2006, pp 115-120, Corte, France.
- Fabrice Bernardi, Thierry Antoine-Santoni. "On the design of object-oriented libraries for systemics". in Proceedings of The 9th World Multi-Conference on Systemics, Cybernetics and Informatics . July 2005. Orlando, Flordia, USA

Poster dans des manifestations nationales

- T. Antoine-Santoni, J.F Santucci, E. De Gentili, B. Costa "Towards an application of Wireless Sensor Network in Wildfire fight. Simulation of routing protocol and deployment strategy". Journée de l'école doctorale de l'Université de Corse. Juin 2007. Corte.
- T. Antoine-Santoni, J.F Santucci, E. De Gentili, B. Costa "Modélisation et simulation d'un réseau de capteurs. Application à la problématique de feux de forets." Présentation lors des Doctoriales du

Grand Sud. Mai 2006. Carry le Rouet. France.

Bibliographie

[Achir et Ouvry, 2005] Achir, M. et Ouvry, L. (2005). Qos and energy consumption in wireless sensor networks using csma/ca. Dans *Proceedings of Systems Communications*, pages 33–39.

[Aiello, 1997] Aiello, A. (1997). *Environnement Orienté Objet de modélisation et de simulation à Evénements Discrets de Systèmes Complexes*. Thèse de Doctorat, Université de Corse.

[Akkaya et Younis, 2006] Akkaya, K. et Younis, M. (2006). A survey of routing protocols in wireless sensor networks. 44 :115–121.

[Akyildiz et al., 2002c] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., et Cayirci, E. (2002c). Wireless sensor networks : a survey. *Computer Networks*, 38(4) :393–422.

[Al-Karaki et Kamal, 2004] Al-Karaki, J. N. et Kamal, A. E. (2004). Routing techniques in wireless sensor networks : a survey. 11 :6–28.

[Al-Karaki et al., 2004] Al-Karaki, J. N., Ul-Mustafa, R., et Kamal, A. E. (2004). Data aggregation in wireless sensor networks - exact and approximate algorithms. Dans *Proceedings of IEEE Workshop on High Performance Switching and Routing (HPSR'04)*, Phoenix, AZ, USA.

[Antoine-Santoni et al., 2007a] Antoine-Santoni, T., Bernardi, F., et Giamarchi, F. (2007a). General methodology for metabolic pathways modeling and simulation using discrete event approach- example on the glycolyis of yeast. Dans *Proceedings of the International Conference on Bioinformatics and Computational Biology (BIOCOMP'07)*, volume 1, pages 58–64, Las Vegas,NV, USA. CSREA Press.

[Antoine-Santoni et al., 2006] Antoine-Santoni, T., Santucci, J. F., Gentili, E. D., et Costa, B. (2006). Using wireless sensor network for wildfire detection. an discrete event approach of an evironmental monitoring tool. Dans *Pro-*

ceedings of First International Symposium on Environment Identities and Mediterranean Area (ISEIM'06), pages 115–120, Corté, France. IEEE.

[Antoine-Santoni et al., 2007b] Antoine-Santoni, T., Santucci, J. F., Gentili, E. D., et Costa, B. (2007b). Discrete event simulation of a routing protocol in wireless sensor network. Dans *Proceedings of theInternational Conference on Modeling, Simulation and Visualization Methods (MSV'07)*, pages 94–100. CSREA Press.

[Antoine-Santoni et al., 2007c] Antoine-Santoni, T., Santucci, J. F., Gentili, E. D., et Costa, B. (2007c). Modeling and simulation oriented components of wireless sensor network using devs formalism. Dans *Proceedings of DEVS Integrative Modelisation and Simulation Symposium 2007(DEVS'07)*. SCS, ACM.

[Antoine-Santoni et al., 2007d] Antoine-Santoni, T., Santucci, J. F., Gentili, E. D., et Costa, B. (2007d). Simulation and visualization method of wireless sensor network performances. Dans *Proceedings of International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'07)*, pages 476–483. SCS, IEEE.

[Benjamin et al., 1993] Benjamin, P., Erratungla, J., Delen, D., et Mayer, R. (1993). Intellignet support for simulation modeling : a description-driven approach. Dans *Proceedings of Summer Computer Simulation Conférence (SCSC'93)*, pages 273–277.

[Bernardi, 2002] Bernardi, F. (2002). *Conception de bibliothèques hiérarchisées de modèles réutilisables selon une approche orientée objet*. Thèse de Doctorat, Université de Corse.

[Blumenthal et al., 2003] Blumenthal, J., Handy, M., Golatowski, F., Haasea, M., et Timmermann, D. (2003). Wireless sensor networks - new challenges in software engineering. Dans *Proceedings of IEEE Conference on Emerging Technologies and Factory Automation (ETFA'03)*, pages 551–556.

[Bolduc et Vangheluwe, 2001] Bolduc, J. S. et Vangheluwe, H. (2001). pythonDEVS : A modeling and simulation package for classical hierarchal DEVS. Rapport technique.

[Braginsky et Estrin, 2002] Braginsky, D. et Estrin, D. (2002). Rumor routing algorithm for sensor networks. Dans *Proceedings of the First Workshop on Sensor Networks and Applications (WSNA'02)*, Atlanta, GA, USA.

[C. Oussalah et Thalens, 1995] C. Oussalah, a. M. M. et Thalens, G. (1995). Emost : un environnement de modélisation par objets de systèmes techniques. 37 :31–44.

[Capocchi, 2005] Capocchi, L. (2005). *Simulation concurrente de fautes comportementales pour des systèmes à événements discrets : Application aux circuits digitaux*. Thèse de Doctorat, Université de Corse.

[Chelius, 2004] Chelius, G. (2004). *Architectures et communications dans les réseaux spontanés sans-fil*. Thèse de Doctorat, Institut National des Sciences Appliquées de Lyon.

[Chen et al., 2002] Chen, B., Jamieson, K., Balakrishnana, H., et Morris, R. (2002). *SPAN : an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks*, volume 8.

[Chen et al., 2003] Chen, M. M., Majidi, C., Doolin, D. M., Glaser, S., et Sitar, N. (2003). Design and construction of a wildfire instrumentation system using networked sensors (poster). Network Embedded Systems Technology (NEST).

[Chiasserini et Garetto, 2004b] Chiasserini, C. et Garetto, M. (2004b). Modeling the performance of wireless sensor networks. Dans *Proceedings of IEEE Infocom'04*.

[Chong et Kumar, 1999] Chong, C. Y. et Kumar, S. P. (1999). Solar-aware routing in wireless sensor networks. *Lecture Notes In Computer Science*, ISSU 2775 :847–852.

[Chong et Kumar, 2003] Chong, C. Y. et Kumar, S. P. (2003). Sensor networks : Evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8) :1247–1256.

[Chu et al., 2002] Chu, M., Haussecker, H., et Zhao, F. (2002). Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks. Dans *The International Journal of High Performance Computing Applications*, volume 16.

[Crossbow-Technology, 2006] Crossbow-Technology (2006). *Wireless Sensor Network Seminar*. Como, Italy.

[Demirkol et al., 2005] Demirkol, I., Ersoy, C., et Alagoz, F. (2005). Mac protocols for wireless sensor networks : a survey. pages 325–349.

[Doolin et al., 2004] Doolin, D. M., Glaser, S. D., et Sitar, N. (2004). Software architecture for gps-enabled wildfire sensorboard. TinyOS Technology Exchange.

[Doolin et Sitar, 2005] Doolin, D. M. et Sitar, N. (2005). Wireless sensor for wildfire monitoring. Dans *Proceedings of SPIE Symposium on Smart Structures and Materials (NDE'05)*, San Diego, CA, USA.

[Downey et al., 2002] Downey, A., Elkner, J., et Meyers, C. (2002). *How to think like a computer scientist, learning with Python*.

[DSN, 1978] DSN (1978). Proceedings of the distributed sensor network workshop (dsn).

[Ermel, 2004] Ermel, E. (2004). *Localisation et Routage géographique dans les réseaux sans fil hétérogènes*. Thèse de Doctorat, Université Paris VI Pierre et Marie CURIE.

[Estrin et al., 1999a] Estrin, D., Govindan, R., Heidemann, J., et Kumar, S. (1999a). Next century challenges : Scalable coordination in sensor networks. Dans *Proceedings of ACM Internatinal Conference Mobile Computing and Networking (MobiCom'99)*, pages 263–270.

[Fang et al., 2003] Fang, Q., Zhao, F., et Guibas, L. (2003). Lightweight sensing and communication protocols for target enumeration and aggregation. Dans *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking and computing (MOBIHOC'03)*, volume 1, pages 165–176.

[Farooq et al., 2004] Farooq, U., Balya, B., et Wainer, G. (2004). Modelling routing in wireless ad-hoc networks using cell-devs. Dans *Proceedings of 2004 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'04)*, pages 285–292, San Jose, CA, USA.

[Fishwick, 1998] Fishwick, P. (1998). issues with web-publishable digital objects. Dans *Proceedings of SPIE Aerosense Conference*.

[Fishwick, 1995] Fishwick, P. A. (1995). *Simulation Model design and execution ; Building digital worlds*. Prentice-Hall editions.

[Gay et al., 2003] Gay, D., Lewis, P., von Behren, R., Welsh, M., Brewer, E., et Culler, D. (2003). The nesc language : a holistic approach to networked embedded systems. Dans *Proceedings of Conference on Programming Language Design and Implementation (PLDI'03)*, San Diego, CA, USA.

[Glaser, 2004] Glaser, S. D. (2004). Some real-world applications of wireless sensor nodes. Dans *Proceedings of SPIE Symposium on Smart Structures and Materials (NDE'04)*, San Diego, CA, USA.

[Hac, 2003] Hac, A. (2003). *Wireless Sensor Designs*. John Wiley and Sons Ltd.

[Heinzelman et al., 2000] Heinzelman, W., Chandrakasan, A., et Balakrishnan, H. (2000). Energy-efficient communication protocols for wireless microsensor networks. Dans *Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS-33)*.

[Heinzelman et al., 1999] Heinzelman, W. R., Kulik, J., et Balakrishnan., H. (1999). Adaptive protocols for information dissemination in wireless sensor networks. Dans *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking*, pages 174–185, Seattle, WA, USA.

[Hill et al., 2000a] Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., et Pister, K. (2000a). System architecture directions for network sensors. Dans *Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'00)*, Cambridge, MA, USA.

[Hill et al., 2000b] Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., et Pister, K. (2000b). System architecture directions for networked sensors. Dans *Proceedings of 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, volume 34, pages 93–104.

[Hohlt et al., 2004] Hohlt, B., Doherty, L., et Brewer, E. (2004). Flexible power scheduling for sensor networks. Dans *Proceedings of Information Processing in Sensor Networks (IPSN'04)*, Berkeley, CA, USA.

[Intanagonwiwat et al., 2000] Intanagonwiwat, C., Govindan, R., et Estrin, D. (2000). Direct diffusion : a scalable and robust communication paradigm for sensor network. Dans *Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom'00)*, Boston, MA, USA.

[Jiao et al., 2005] Jiao, B., Son, S., et Stankovic, J. (2005). Gem : Generic event service middleware for wireless sensor networks. Dans *Proceedings of International Workshop on Networked Sensing System (INSS'05)*.

[Kim, 2006b] Kim, T. (2006b). Devs-ns2 environment ; an iinteagrated tool for efficient networks modeling and simulation. Dans *Master Thesis, University of Arizona*, AZ, USA.

[Kuhn et al., 2003] Kuhn, F., Wattenhofer, R., et Zollinger, A. (2003). Worst-case optimal and average-case efficient geometric ad-hoc routing. Dans *Proceedings of the 4th ACM International Conference on Mobile Computing and Networking*, pages 267–278.

[Lafarge, 2006] Lafarge, E. (2006). Evaluation des dispositifs de détection des feux de forêts en france. Mémoire de Master, Ecole Nationale du Génie Rural et des Eaux et des Forêts.

[Levis et al., 2003] Levis, P., Lee, N., Welsh, M., et Culler, D. (2003). Tossim : accurate and scalable simulation of entire tinyos applications. Dans *Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys'03)*, pages 126–137, Los Angeles, CA, USA. ACM Press.

[Li et al., 2001] Li, Q., Aslam, J., et Rus, D. (2001). Hierarchical power-aware routing in sensor networks. Dans *Proceedings of the DIMACS Workshop on Pervasive Networking*.

[Li et al., 2006] Li, Y., Wang, Z., et Song, Y.-Q. (2006). Wireless sensor network design for wildfire monitoring. Dans *Proceedings of 6th World Congress on Intelligent Control and Automation (WCICA'06)*, pages 109–113, Dalia, China. IEEE.

[Lindsey et Raghavendra, 2002] Lindsey, S. et Raghavendra, C. (2002). Pegasis : Power-efficient gathering in sensor information systems. Dans *Proceedings of IEEE Aerospace Conference*, volume 3, pages 9–16.

[Liu et al., 2001] Liu, J., Perrone, L. F., Nicol, D. M., et Liljenstam, M. (2001). Simulation modeling of large-scale ad-hoc sensor networks. Dans *Proceedings of the 2001 Simulation Interoperability Workshop*, London, England.

[Manjeshwar et Agarwal, 2001] Manjeshwar, A. et Agarwal, D. P. (2001). Teen : a routing protocol for enhanced effeciency in wireless sensor networks. Dans *Proceedings 1st International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*.

[Manjeshwar et Agerwal, 2002] Manjeshwar, A. et Agerwal, D. P. (2002). Apteen : A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks. Dans *Proceedings of Parallel and Distributed Processing Symposium (IPDPS'02)*, pages 195–202.

[Maurice, 2005] Maurice, R. (2005). *Contribution à la Méthodologie de Conception de Système : Application à la Réalasiation d'un Micorsystème Multicapteurs Communicant pour le Génie Civil*. Thèse de Doctorat, Laboratoire d'Analyse et d'Arcitecture des Systèmes du CNRS de Toulouse.

[Muller et Gaertner, 2000] Muller, P. et Gaertner, N. (2000). *Modélisation Objet avec UML, Deuxième Edition*. Eyrolles.

[Muzy, 2004] Muzy, A. (2004). *Elaboration de modèles déterministes par la simulation de systèmes spatiaux complexes : application à la problématique des feux de forêts*. Thèse de Doctorat, Université de Corse.

[NS2, 1995] NS2 (1995). Available : http ://www.isi.edu/nsnam/ns/.

[Omnet++, 1992] Omnet++ (1992). Available : http ://www.omnetpp.org.

[Oussalah, 1998] Oussalah, C. (1998). Modèles hiérarchisés multi-vues pour le support de raisonnement dans les domaines techniques. Rapport technique, Université d'Aix-Marseille.

[Oussalah, 1999] Oussalah, M. (1999). *Génie Objet, Analyse et Conception de l'évolution*. Editions Hermès.

[Park et al., 2000] Park, S., Savvides, A., et Srivastava, M. B. (2000). Sensorsim : a simulation framework for sensor networks. Dans *Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems (MSWIM'00)*, pages 104–111, Boston, Massachusetts, United States. ACM Press.

[Park et al., 2001] Park, S., Savvides, A., et Srivastava, M. B. (2001). Battery capacity measurement and analysis using lithium coin cell battery. Dans *Proceedings of the 2001 international symposium on Low power electronics and design(ISLPED'01)*, pages 382–387, New York, NY, USA. ACM Press.

[Polley et al., 2004] Polley, J., Blazakis, D., Mcgee, J., Rusk, D., et Baras, J. S. (2004). Atemu : a fine-grained sensor network simulator. Dans *Proceedings of IEEE Conference on Sensor and Ad Hoc Communications and Networks (SECON'04)*, pages 145–152.

[Popper, 1973] Popper, J. (1973). *La Dynamique des Systèmes, Principes et Applications.* Les Editions d'Organisation.

[Prométhée, 1973] Prométhée (1973). La banque de données sur les incendies de forêts en région méditerranéenne en france. http ://www.promethee.com.

[Rodoplu et Meng, 1999] Rodoplu, V. et Meng, T. H. (1999). Minimum energy mobile wireless networks. 17.

[Sadagopan et al., 2003] Sadagopan, N., Krishnamachari, B., et Helmy, A. (2003). The acquire mechanism for efficient querying in sensor networks. Dans *Proccedings of the First IEEE International Workshop on Sensor Network Protocols and Applications (SNPA'03)*, Anchorage, AK, USA.

[Samper et al., 2006] Samper, L., Maraninchi, F., Mounier, L., et Mandel, L. (2006). Glonemo : global and accurate formal models for the analysis of ad-hoc sensor networks. Dans *Proceedings of the first international conference on Integrated internet ad hoc and sensor networks(InterSense'06)*, page 3, Nice, France. ACM Press.

[Savvides et al., 2001] Savvides, A., Park, S., et Srivastava, M. B. (2001). On modeling networks of wireless microsensors. Dans *Proceedings of the 2001 ACM SIGMETRICS international conference on Measurement and modeling of computer systems(SIGMETRICS'01)*, pages 318–319, Cambridge, MA, USA. ACM Press.

[Schurgers et Srivastava, 2001] Schurgers, C. et Srivastava, M. (2001). Energy efficient routing in wireless sensor networks. Dans *MILCOM Proceedings on Communication for Network-Centric Operations : Creating the Information Force*, McLean, VA, USA.

[Servetto et Barrenechea, 2002] Servetto, S. et Barrenechea, G. (2002). Constrained random walks on random graphs : Routing algorithms for large scale wireless sensor networks. Dans *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, volume 16, Atlanta, GA, USA.

[Shah et Rabaey, 2002] Shah, R. et Rabaey, J. (2002). Energy aware routing for low energy ad hoc sensor networks. Dans *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC'02)*, Orlando, FL, USA.

[Shnayder et al., 2004] Shnayder, V., Hempstead, M., Chen, B. R., Allen, G. W., et Welsh, M. (2004). Simulating the power consumption of large-scale sensor network applications. Dans *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys'04)*, pages 188–200, Baltimore, MD, USA. ACM Press.

[SOS, 2004] SOS (2004). the sos operating system. Available : http ://nesl.ee.ucla.edu/projects/sos/.

[Stein et Loucas, 1995] Stein, J. et Loucas, L. (1995). A component-based modeling approach for system design : Theory and implementation. Dans *Proceedings of International Conference of Bond Graph Modeling (ICBGM'95)*.

[Stojmenovic et Lin, 1999] Stojmenovic, I. et Lin, X. (1999). Gedir : Loop-free location based routing in wireless networks. Dans *Proceedings of the International Conference on Parallel and Distributed Computing and Systems*, pages 267–278, Boston, MA, USA.

[Subramanian et Katz, 2000] Subramanian, L. et Katz, R. H. (2000). An architecture for building self configurable systems. Dans *Proceedings of IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing*, volume 1, Boston, MA, USA.

[Sundresh et al., 2004] Sundresh, S., Kim, W., et Agha, G. (2004). Sens : A sensor, environment and network simulator. Dans *Proceedings of the 37th annual symposium on Simulation (ANSS'04)*, page 221, Washington, DC, USA. IEEE Computer Society.

[Swinnen, 2003] Swinnen, G. (2003). Apprendre à programmer avec python.

[Titzer, 2004] Titzer, B. L. (2004). *Avrora : The AVR Simulation and Analysis Framework*. Thèse de Doctorat, University of California, Los Angeles.

[Titzer et al., 2005] Titzer, B. L., Lee, D. K., et Palsberg, J. (2005). Avrora : scalable sensor network simulation with precise timing. Dans *Proceedings Fourth International Symposium on Information Processing in Sensor Networks*, pages 477–482, Los Angeles, CA, USA.

[Warneke et al., 2001] Warneke, B., M. Last, a. B. L., et Pister, K. (2001). Smart dust : Communicating with a cubic-millimeter computer. Dans *Computer*, volume 34, pages 44–51.

[Wise et Najafi, 1991] Wise, K. et Najafi, N. (1991). The coming opportunities in microsensor systems. Dans *Solid-State Sensors and Actuators. Digest of Technical Papers, TRANSDUCERS*, pages 2–7.

[Woo et al., 2003a] Woo, A., Tong, T., et Culler, D. (2003a). Taming the underlying challenges of reliable multihop routing in sensor networks. Dans *Proceedings of the First International Conference on Embedded Networked Sensor Systems (SenSys'03)*.

[Xu, 2002] Xu, N. (2002). A survey of sensor network applications. *IEEE Communications Magazine*, pages 102–114.

[Xu et al., 2001] Xu, Y., Heidemann, J., et Estrin, D. (2001). Geography-informed energy conservation for ad-hoc routing. Dans *Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 70–84.

[Yao et Gehrke, 2002] Yao, Y. et Gehrke, J. (2002). *The cougar approach to in-network query processing in sensor networks*.

[Ye et al., 2003] Ye, F., Chen, A., Liu, S., et Zhang, L. (2003). A scalable solution to minimum cost forwarding in large sensor networks. Dans *Proceedings of the tenth International Conference on Computer Communications and Networks (ICCCN'03)*, pages 304–309.

[Ye et al., 2002a] Ye, F., Luo, H., Cheng, J., Lu, S., et Zhang, L. (2002a). A two-tier data dissemination model for large-scale wireless sensor networks. Dans *Proceedings of ACM/IEEE MOBICOM*, volume 1.

[Ye et al., 2002b] Ye, W., Heidemann, J., et Estrin, D. (2002b). Data aggregation in wireless sensor networks - exact and approximate algorithms. Dans *Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'02)*, New York, NY, USA.

[Yu et al., 2001] Yu, Y., Estrin, D., et Govindan, R. (2001). Geographical and energy-aware routing : A recursive data dissemination protocol for wireless sensor networks. Rapport technique.

[Zeigler, 1976] Zeigler, B. P. (1976). *Theory of Modeling and Simulation*. Academic Press.

[Zeigler, 1990] Zeigler, B. P. (1990). *Object-Oriented Simulation with Hierarchical, Modular Models*. Academic Press.

[Zeigler et al., 2000] Zeigler, B. P., Praehofer, H., et Kim, T. G. (2000). *Theory of Modeling and Simulation, Second Edition*. Academic Press.

[Ácc et Buttyán, 2006] Ácc, G. et Buttyán, L. (2006). A taxonomy of routing protocols for wireless sensor networks. Dans *Híradástechnika*.

Liste des figures

Liste des tableaux

Liste des algorithmes

Annexe 1 : Simulation avec DEVS-WSN

Dans ce chapitre, nous allons montrer deux exemples du déroulement de la simulation avec notre application **DEVS-WSN**. Nous ne disposons pas encore d'environnement graphique, pour visualiser le réseau de capteurs. Les résultats que nous obtenons, sont générés à partir de la station de base sous forme de fichier .xls par le biais d'un script python pyXLWriter. Cependant nous proposons d'étudier le comportement de **DEVS-WSN**, et pour cela nous procédons à deux captures d'écran, que nous détaillons.

Visualisation de l'échange de message

Nous détaillons, dans cette partie, le processus permettant l'échange d'un message de type ACK entre le noeud 2 (node 2) et le noeud 1 (node 1) illustrée par la Figure 6.13.

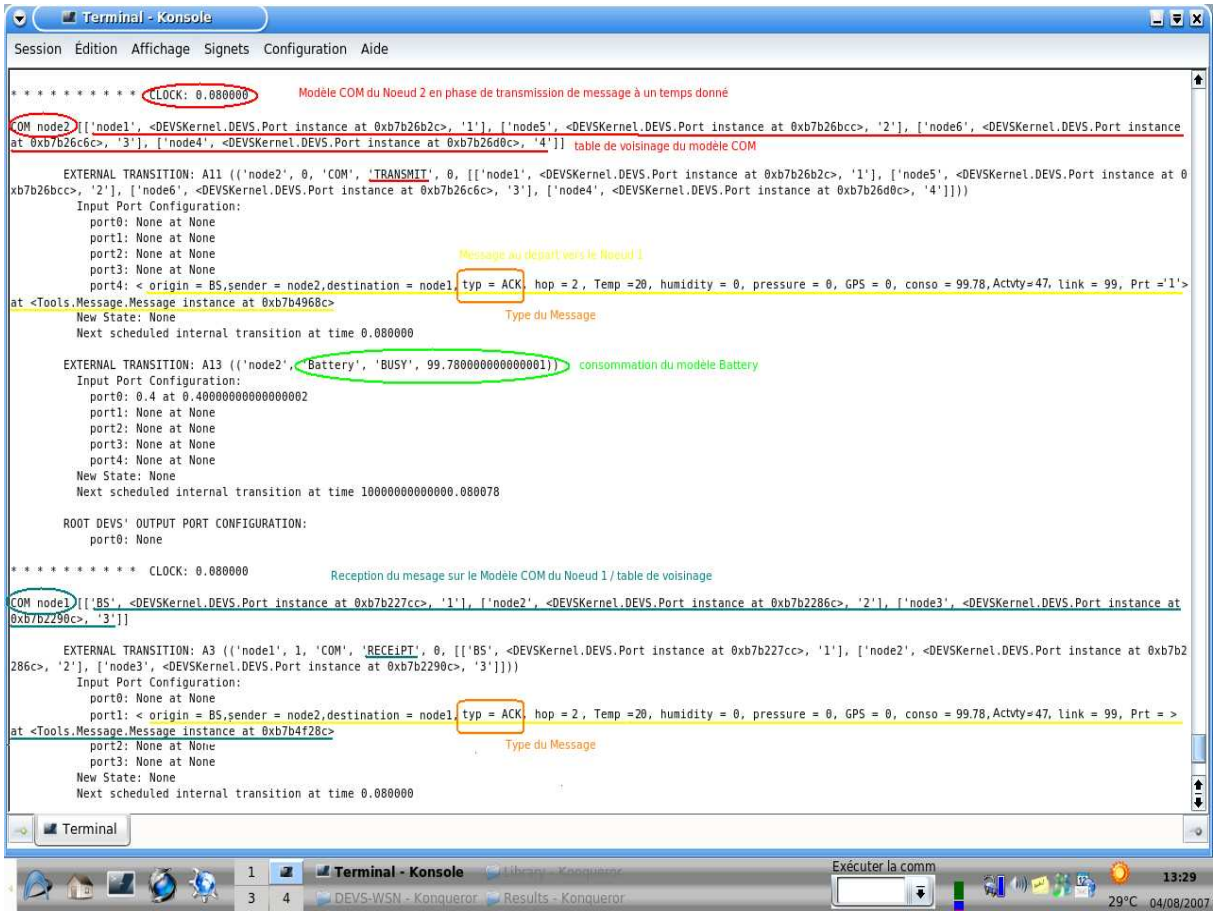


FIG. 6.13*: Echange de messages entre deux noeuds*

Nous observons le déroulement d'une phase de communication entre les deux modèles "*COM*" des noeuds 1 et 2. En rouge nous faisons apparaître le nom du modèle et la table voisinage. La table de voisinage (*cf 4.1.3.2*) recence toutes les connections existantes entre un noeud et ses voisins. Cette table est alimentée au fur et à mesure de la connection de nouveaux noeuds. Elle ne détermine en aucun cas le routage qui lui est précisé par la table de routage au sein du MA "*Net*". Nous voyons que le MA "*COM*" est en cours de transmission de message, identifiable par la phase ***'TRANSMIT'***. Nous faisons apparaître en jaune le départ d'un message sur le port 4, à destination du noeud 1. Nous distinguons le type de message, identifiable par le terme "ACK", qui indique que ce message est un message d'acquittement pour signifier, dans ce cas précis une fin de communication. Le cercle vert a pour but de mettre en relief la diminution de la capacité énergétique au cours de la simulation, au moyen du MA "*Battery*". En bleu, nous mettons en

évidence la table de voisinage du noeud 1 qui est naturellement différente le MA "*COM*" du noeud 1 qui est en phase ***'RECEiPT'***.

Visualisation de la mise à jour de la table de routage

Nous proposons de visualiser le comportement du MA "*Net*", en dsitinguant sa table de routage. Nous représentons ce processus sur la Figure 6.14.

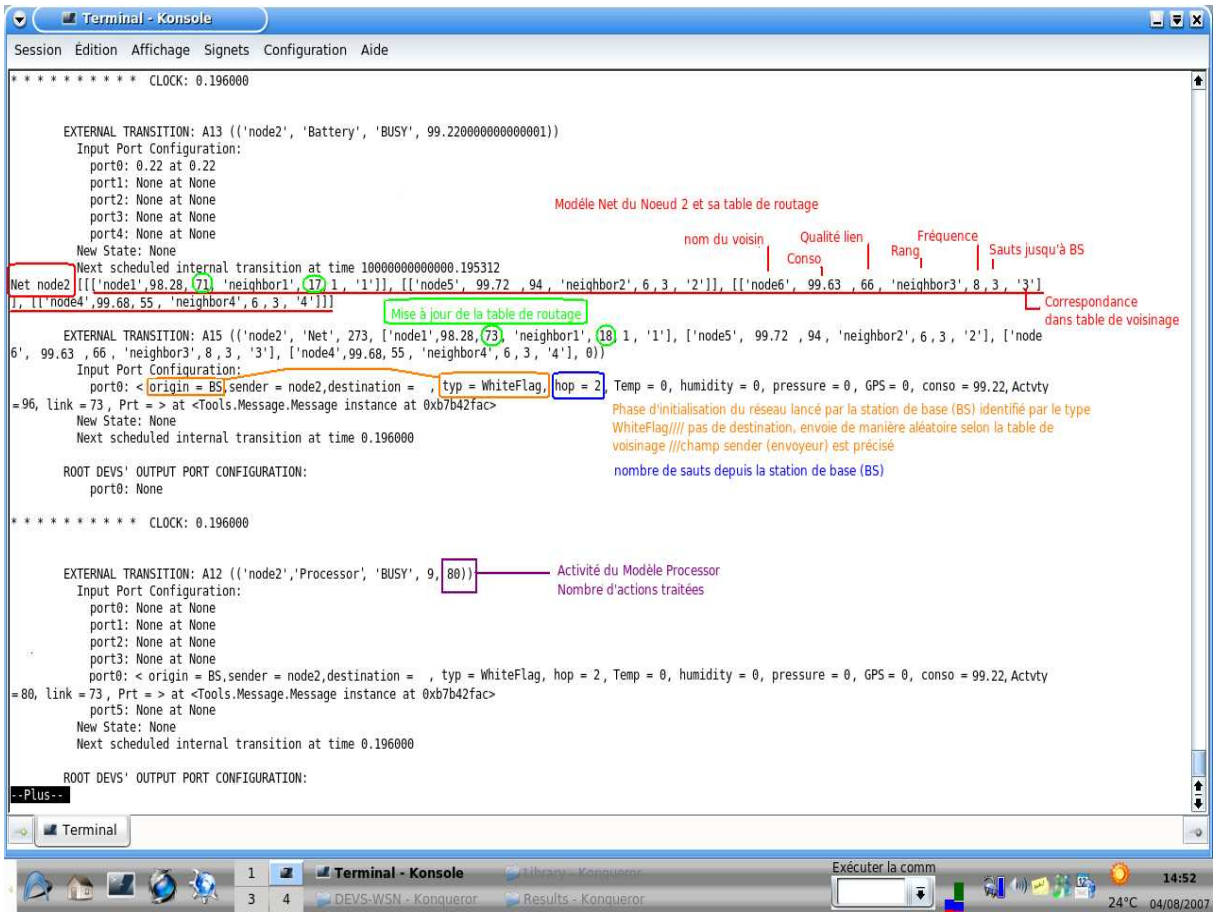


FIG. 6.14*: Mise à jour de la table de routage*

Nous identifions sur cette figure, la table de routage du noeud 2. Cette dernière est représentée par différents champs :

– nom du noeud ;

- valeur énergétique du noeud, envoyée lors de la dernière transmission de message ;
- qualité de lien ;
- rang qui identifie la position du noeud dans la hiérarchie de la table ;
- fréquence qui identifie le nombre de messages reçus par le noeud 2 en fonction du noeud considéré ;
- sauts qui représente le nombre de sauts pour atteindre la station de base ;
- la correspondance représente le lien qui existe entre la table de voisinage et la table de routage ; par exemple nous savons que le message sortant du MA "*Net*" avec le port '3' comme destination , trouvera son port correspondant dans la table de voisinage du MA "*COM*". Dans notre exemple, c'est un message de diffusion et de mise à jour du réseau de type "WhiteFlag" avec pour origine la station de base. Ce message n'a pas besoin de port, et sera envoyé de manière aléatoire sur un port de sortie du MA "*COM*".

Nous faisons apparaître en violet le MA "Processor" qui incrémente la valeur activité au fur et à mesure de la simulation.

Annexe 2 : Tutoriel du logiciel MOTE-VIEW

Nous présentons le logiciel qui va permettre le déploiement d'un réseau de capteurs sans fil, de type MICA2. La société Crossbow Technology fournit dans cette optique, le logiciel MOTE-VIEW qui est l'interface entre l'utilisateur et le réseau déployé de capteurs sans fil. Ce logiciel fournit les outils pour simplifier le déploiement et la surveillance du réseau. Il simplifie également la connection à une base de données et l'analyse des données envoyées par les capteurs. Nous présentons dans un premier temps la phase d'installation de MOTE-VIEW et ensuite nous détaillons la préparation des capteurs MICA2 avant le déploiement du réseau.

Phase d'installation du logiciel

Prérequis

Les droits d'administrateurs sont requis pour l'installation du logiciel MOTE-VIEW. Le logiciel MOTE-VIEW est supporté uniquement par les plates-formes : Windows XP Home, Windows XP Professionel, Windows 200 avec SP4. Tous les outils de visualisation inclus dans Mote-View nécessitent d'être connectés à une base de données. Cette base de données peut être installée sur le PC (en tant que "localhost") ou sur un serveur distant. Des logiciels complémentaires sont donc nécessaires pour une utilisation du logiciel MoteView :

– PostgreSQL 8.0 service de base de données ;

– le pilote PostgreSQL ODBC ;

– le cadre d'application Microsoft.Net

L'installation de la PostgreSQL et la configuration de la base de données locale PostgreSQL 8.0 se font automatiquement lors de l'installation de MOTE-VIEW, en validant l'option d'installation automatique de ces composants.

Installation rapide

Nous décrivons dans cette partie la phase d'installation du logiciel MOTE-VIEW et de ses composants :

1. Insertion du CD *Wireless Sensory Systems Support ;*
2. Double-clic sur ***MoteViewSetup.exe*** à partir du dossier MOTE-VIEW ;
3. Arrivée à la fenêtre de bienvenue de MOTE-VIEW, un clic sur **Next>** ;
4. Sélection du répertoire d'installation, puis cliquer sur **Next>** comme représenté sur la Figure 6.15 :

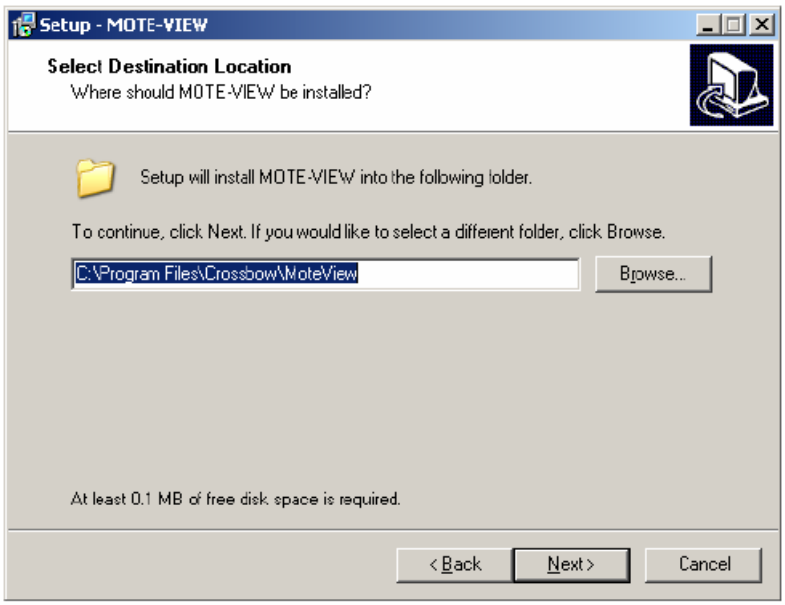


FIG. 6.15*: Sélection du dossier d'installation*

5. Sélection du nom du dossier de démarrage puis cliquer sur **Next>** comme sur la Figure **6.16** ;

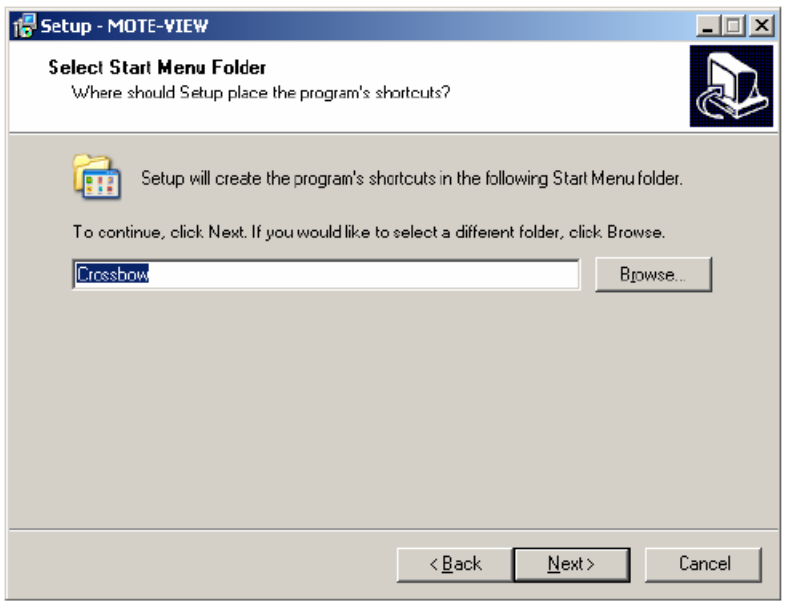


FIG. 6.16*: Sélection du dossier pour le menu de démarrage de MOTE-VIEW*

6. Sélection des composants[4] nécessaires à l'installation puis cliquer sur **Next>** comme sur la Figure **6.17** ;

[4]Sélectionner toutes les otpions si c'est la première installation

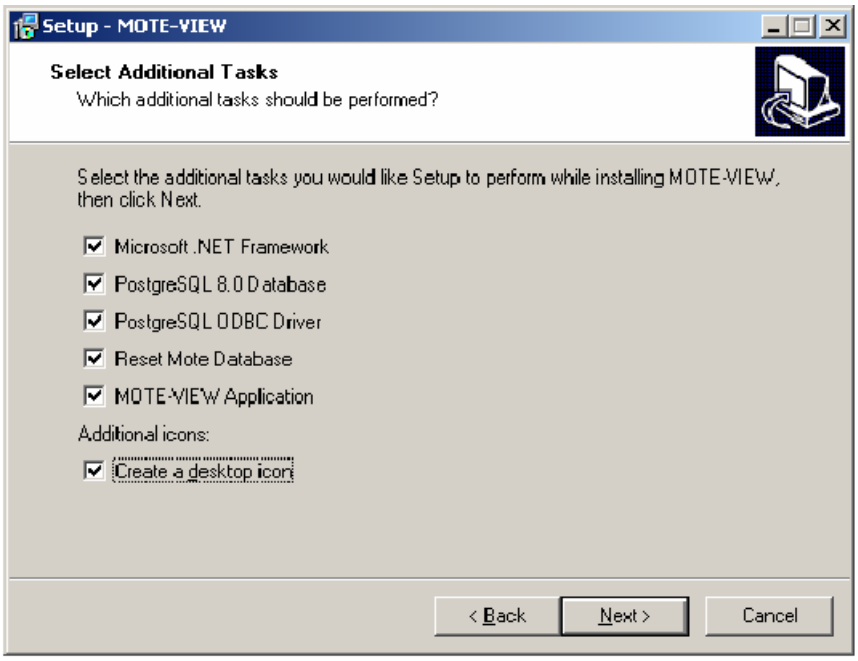


FIG. 6.17*: Sélection des composants requis pour MOTE-VIEW*

7. Confirmations des sélections et cliquer sur **Install** comme sur la Figure 6.18.

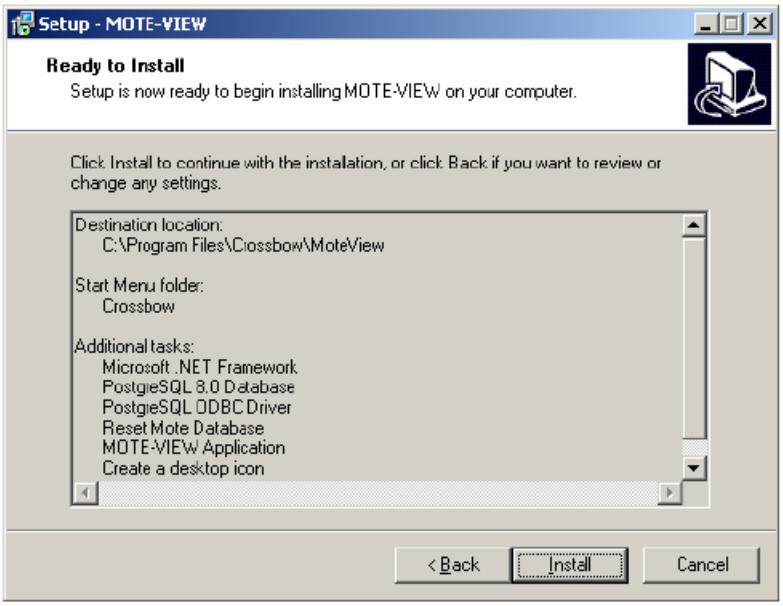


FIG. 6.18*: Confirmation des sélections*

Barre de menu du logiciel

Maintenant que MOTE-VIEW est installé, il est nécessaire de détailler l'application. Pour cela nous étudions la barre de menu représentée par la Figure 6.19.

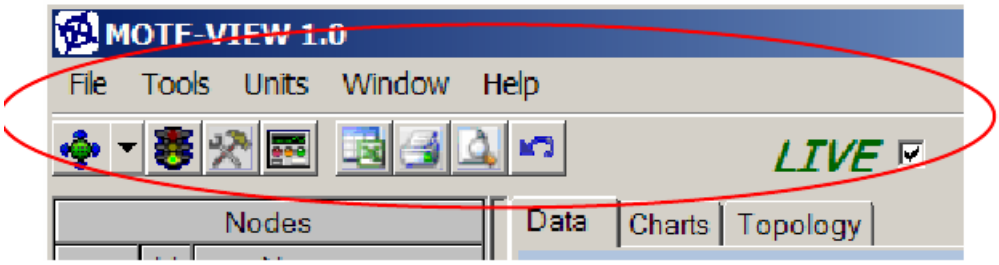


FIG. 6.19*: Barre de menu de MOTE-VIEW*

L'icône vert, sur la droite, permet d'ouvrir la fenêtre pour la connection au serveur (local ou distant) et à la base de données. Le nom de la table va dépendre de l'application des capteurs.

L'icône, représentant un feu vert, permet la connection entre la station d'acquisition (station de base) et le réseau de capteurs déployés. On définit également la plate-forme à utiliser (MICA2 par exemple) et l'application, c'est à dire le type de carte "capteurs" (XMTS420 pour la MTS420 entre autres). Un bouton start permet de lancer le processus de connections.

L'icône représentant un marteau et une clé permet de lancer la fenêtre pour la configuration des capteurs. On y définit le type d'application, les fréquences et les identifiants des noeuds.

Le triangle jaune ouvre le panneau de configuration de l'alerte. Par ce panneau, il est possible de créer des messages d'alertes en fonction des données recueillies. Si par exemple, la température collectée est supérieure par exemple à 30 ˚C , le logiciel peut générer un message de type "pop-up" ou "e-mail", signifiant à l'utilisateur, que ce seuil a été franchi.

L'icône, représentant une console, autorise la configuration de certains paramètres de MOTE-VIEW, tel que le rafraîchissement de la page ou la résolution des graphiques.

Les autres icônes permettent de réaliser des exportations de données en format .csv ou.xml, une impression et un aperçu avant impression de manière assez classique.

L'option ***LIVE*** permet d'avoir des graphiques au fur à mesure de l'arrivée des données sur la station de base.

Ecran de contrôle

MOTE-VIEW fournit un écran de contrôle qui permet de visualiser le réseau de capteurs déployé sur le terrain. Nous représentons sur la Figure 6.20 une capture d'écran de l'interface proposée par MOTE-VIEW.

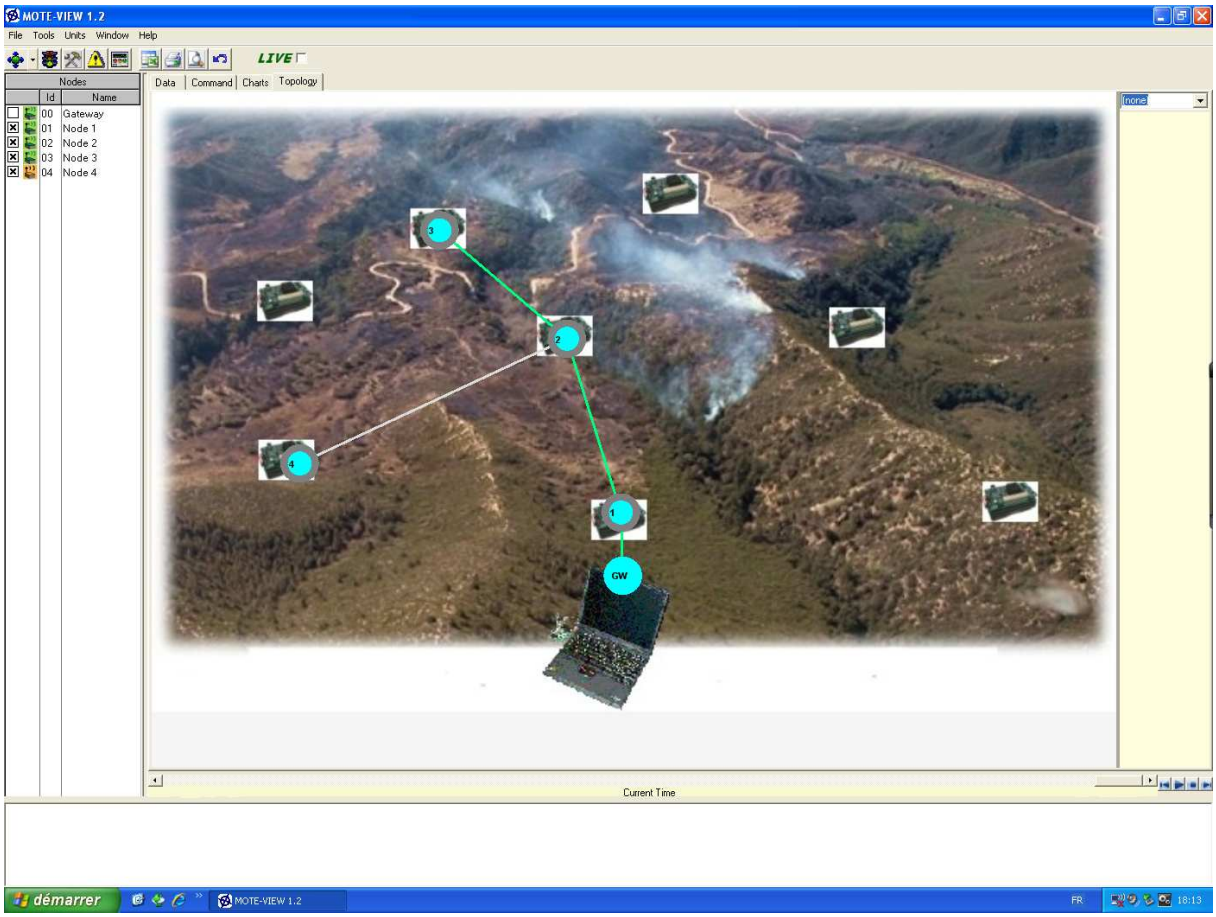


FIG. 6.20*: Interface de contrôle du logiciel MOTE-VIEW*

Nous pouvons distinguer à droite la zone **Nodes**, permettant d'indentifier les capteurs présents sur le réseau et leur état. Au centre, nous voyons la fenêtre principale qui se décompose en plusieurs onglets : l'onglet **Data** permettant de visualiser l'arrivée des données sur la station de base ; l'onglet **Command** autorisant des tests techniques sur les capteurs au moyen de diodes présentes sur la plate-forme MICA2 ; l'onglet **Chart** qui trace au fur et à mesure de la collecte des informations, les courbes relatives aux différents paramètres des cartes "capteurs" (température, pression,

accélération, lumnosité, etc.) ; l'onglet **Topology** permet de visualiser la relation des capteurs en visualisant les liens entre ces derniers comme illustré sur la Figure **6.20** ; le lien vert entre deux capteurs signifie que ces derniers ont une relation priviligiée, c'est à dire il représente le vosin idéal de chaque noeud pour atteindre la station de base.

Nous montrons sur le Tableau 6.2 , les caractéristiques de la zone **Node**, permettant d'observer certaines caractéristiques des noeuds.

Options dans la zone liste	Description
Cocher la case du Noeud	Le graphique des paramètres de la carte "capteurs" est tracé
Changer les propriétés du Noeud	Double clic dans la liste des noeuds ou clic droit (choisir propriétés)
Trier les données	Click on the column header. Static display only when not in "live" mode.
Ajouter des noeuds	Clic droit sur un noeud ou alors utiliser l'outil d'auto-détectection
 Icône capteur grise	Aucune information n'est reçue
 Icône capteur verte	Raffraîchissement des données inférieurs à 20 minutes
 Icône capteur vert clair	Raffraîchissement des données > 20 minutes
 Icône capteur jaune	Raffraîchissement des données > 40 minutes
 Icône capteur orange	Raffraîchissement des données > 60 minutes
 Icône capteur rouge	Raffraîchissement des données > 1 jour

TAB. 6.2*: Caractéristiques de la zone **Nodes** de MOTE-VIEW*

Cette zone est importante car elle permet d'identifier les possibles problèmes de communication et de non-récéption des données par le biais d'icônes de couleur.

En bas de l'écran de contrôle, nous trouvons la boîte de dialogue du serveur, permettant de suivre les différentes opérations de mise en place.

Préparation des capteurs

Nous venons de voir les caractéristiques du logiciel MOTE-VIEW. Nous nous intéressons à présent à la phase de préparation des capteurs. Cette phase a pour objectif de charger les applications logiciels de TinyOS, pré-compilés, au capteur. Pour bien comprendre cette phase, nous

utiliserons cinq capteurs MICA2 et de quatres cartes ”capteurs” MTS420 et une station de base MIB510 (*cf* 6.2). Ainsi nous aurons les capteurs 1, 2, 3, 4 déployés sur le terrain et le capteur 0 représentant l'entité liée à la station de base. Dans la phase de préparation, le capteur 0 est toujours préparé en dernier. Dans notre exemple, nous allons détailler la phase de programmation du capteur 2. Pour autoriser cette phase de péparation, il est nécessaire de coupler le capteur à la station de base comme illustré sur la Figure 6.21.

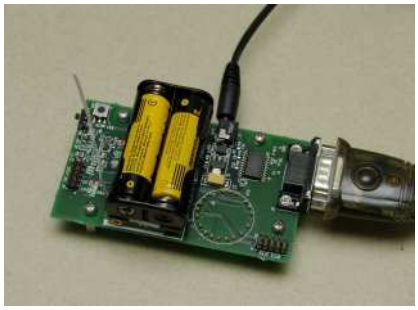


FIG. 6.21*: Couplage MICA2 et station de base MIB510*

Après ce couplage, il faut charger les applications qui vont être utilisées par le capteur. Cette opération se réalise grâce à la boite de dialogue MoteConfig, dont le raccourci est l'îcone représentée par le marteau et la clé. Le but de cette préparation est de configurer les entités du réseau afin de leur fournir les applications et les paramètres nécessaires pour analyser correctement les données envoyées par ces dernières. Nous représentons sur la Figure 6.22, la fenêtre de configuration MoteConfig.

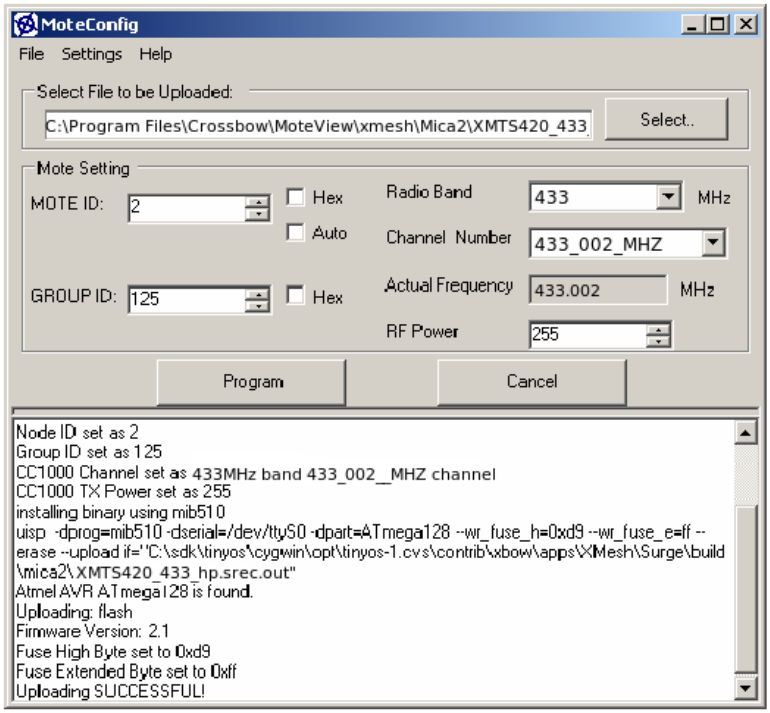


FIG. 6.22*: La fenêtre MoteConfig*

Si nous détaillons la fenêtre MoteConfig, la première étape consiste à choisir l'application qui sera chargé dans chaque capteur. Ici, nous possédons des capteurs MICA2 avec une carte MTS420, que nous utilisons sur la fréquence 433 MHz. Une fois cette étape passée, nous définisons en conséquence du choix de l'application, les caractéristiques de fréquence : nous choisissons la fréquence 433 MHz avec une puissance d'émission maximale fixé à 255 MHZ.

La troisième étape consiste à identifier les noeuds. Ce processus a pour objectif de donner des noms à des noeuds et à des groupes de noeuds. Ces noms seront seulement utiliser pour identifier la source émettrice lors de la réception des données sur la station de base.

Une fois ces différentes étapes passées, un clic sur **Program** permet de charger les applications sur le capteur. Si cette étape est réussie, un message "Uploading SUCCESSFUL !" est renvoyé. Dans le cas où par exemple le capteur est mal couplé à la station de base, le message "Uploading FAILED !" apparaît.

Annexe 3 : Autres résultats expérimentaux

Nous proposons dans cette partie d'autres résultats issus de l'expérimentation de Noceta. Ces résultats ne présentent pas de réels apports face au problème étudié, cependant nous choisissons de les fournir pour apporter une vue plus exhaustive sur les données recueillies dans le cadre expérimental.

Visualisation à l'aide du logiciel Mote-View

Durant la phase d'expérimentation, nous avons procédé à une capture d'écran pour illustrer le logiciel MoteView. Notre dispositif, constitué de la station de base, du Noeud 0 et de l'ordinateur portable, était disposé assez près du feu expérimental. Le matériel était protégé mais il était difficile pour nous de pouvoir rester à côté vu la quantité de chaleur dégagée et la fumée à ce moment là. Cependant nous avons réussi à réaliser une capture d'écran représentée sur la Figure 6.23.

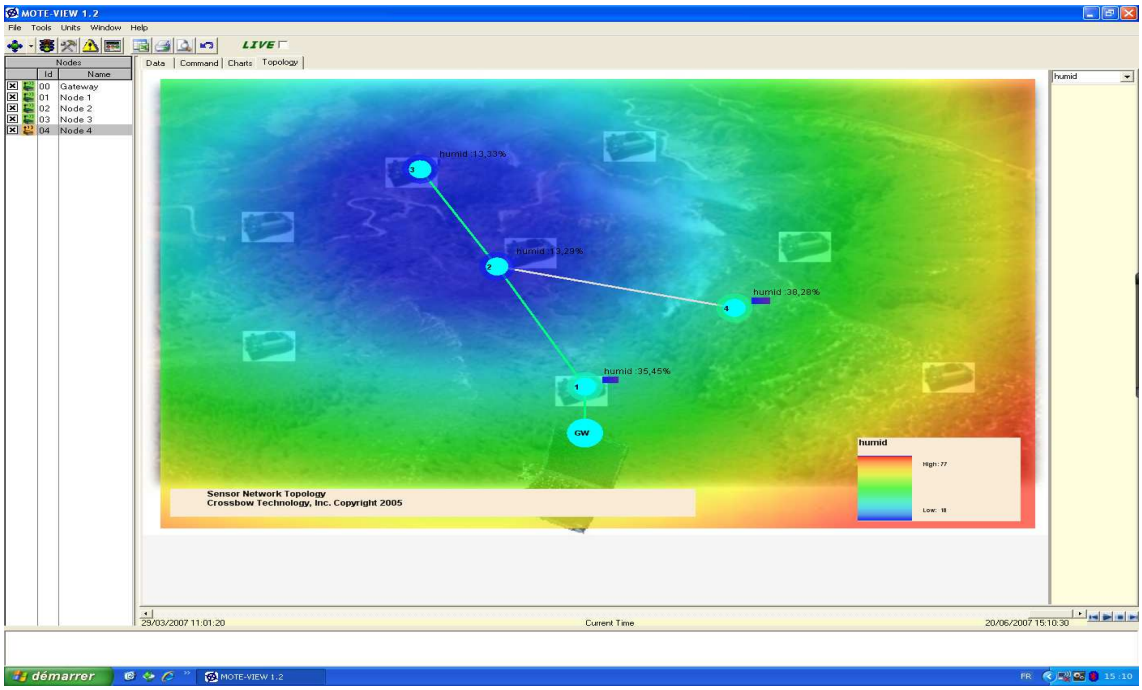


Figure 6.23*: Visualisation du taux d'humidité selon Moteview*

Nous observons le taux d'humidité en fin d'expérimentation par le biais de l'interface MOTE-VIEW. Nous voyons de manière claire la zone brûlée, en bleu. En effet, nous avons vu dans les résultats de l'expérimentation (*cf* 6.4) , qu'après le passage du feu, le taux d'humidité a chuté de manière significative. Cette représentation assez ludique offre une solution différente et rapide d'observation d'un phénomène environnemental.

Autres résultats

Positions rapportées par le module GPS

Les modules GPS intégrés dans les cartes "capteurs" MTS420 ont fourni les coordonnées géographiques des différents noeuds, référencées dans le Tableau 6.3.

	Latitude	***Longitude***
Noeud 1	N42, 19, 538	E9, 20, 406
Noeud 2	N42, 19, 540	E9, 20, 390
Noeud 3	N42, 19, 547	E9, 20, 378

TAB. 6.3*: Données GPS de la plate-forme expérimentale*

Ces données géographiques permettent de localiser de manière relativement précise le feu.

Pression atmosphérique et vibrations

Nous présentons des résultats complémentaires sur la pression atmosphérique, sur la Figure 6.24, et les vibrations sur la Figure 6.25, concernant la plate-forme expérimentale de Noceta. Les résultats sur les vibrations (accéléromètre) ne présentent pas d'indications réelles sur la présence du feu. Seule la pression atmosphérique montre un phénomène naissant de feu, mais le manque de précision des données nous empêchent de les interpréter.

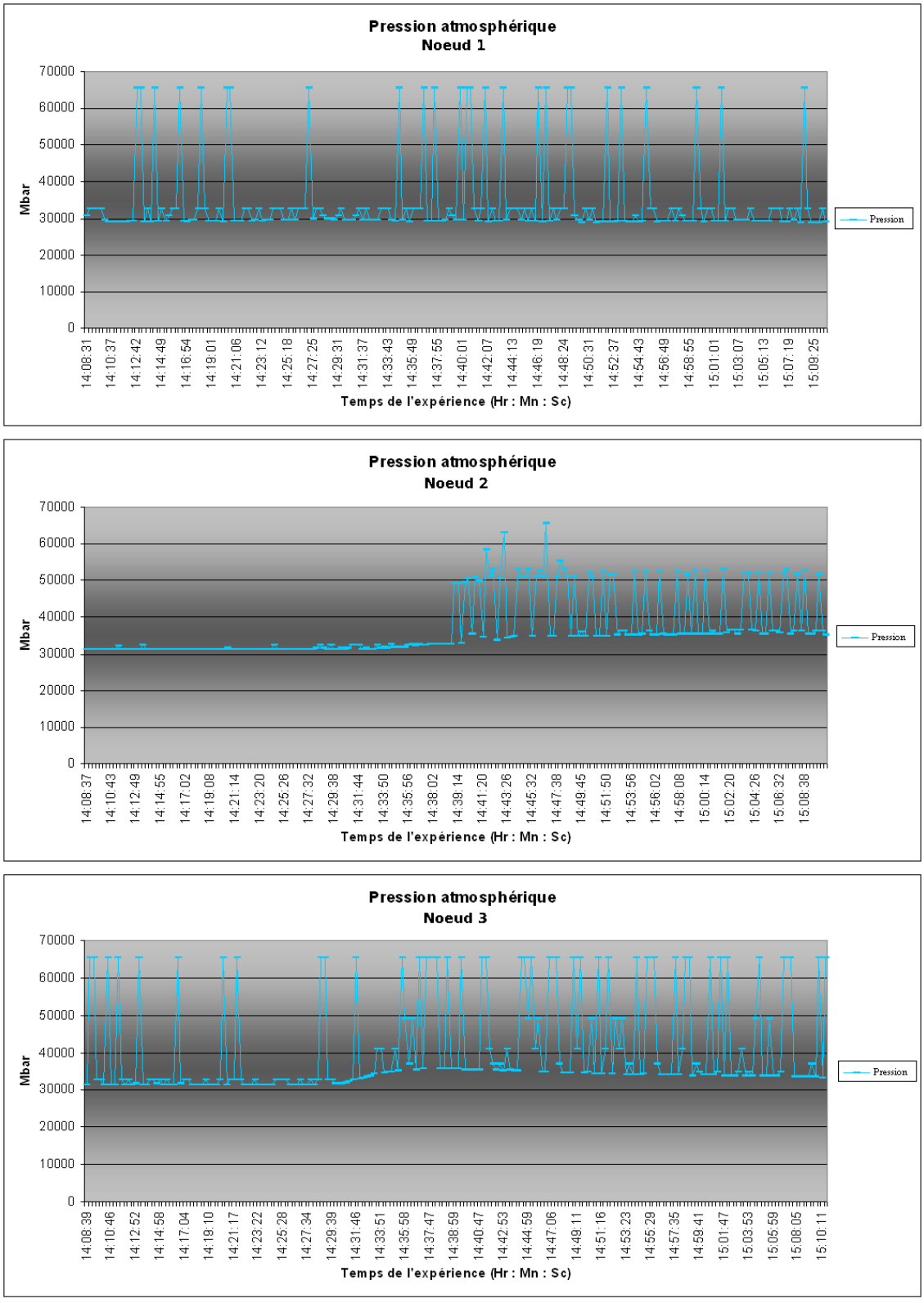


Figure 6.24*: Pression atmosphérique durant l'expérimentation*

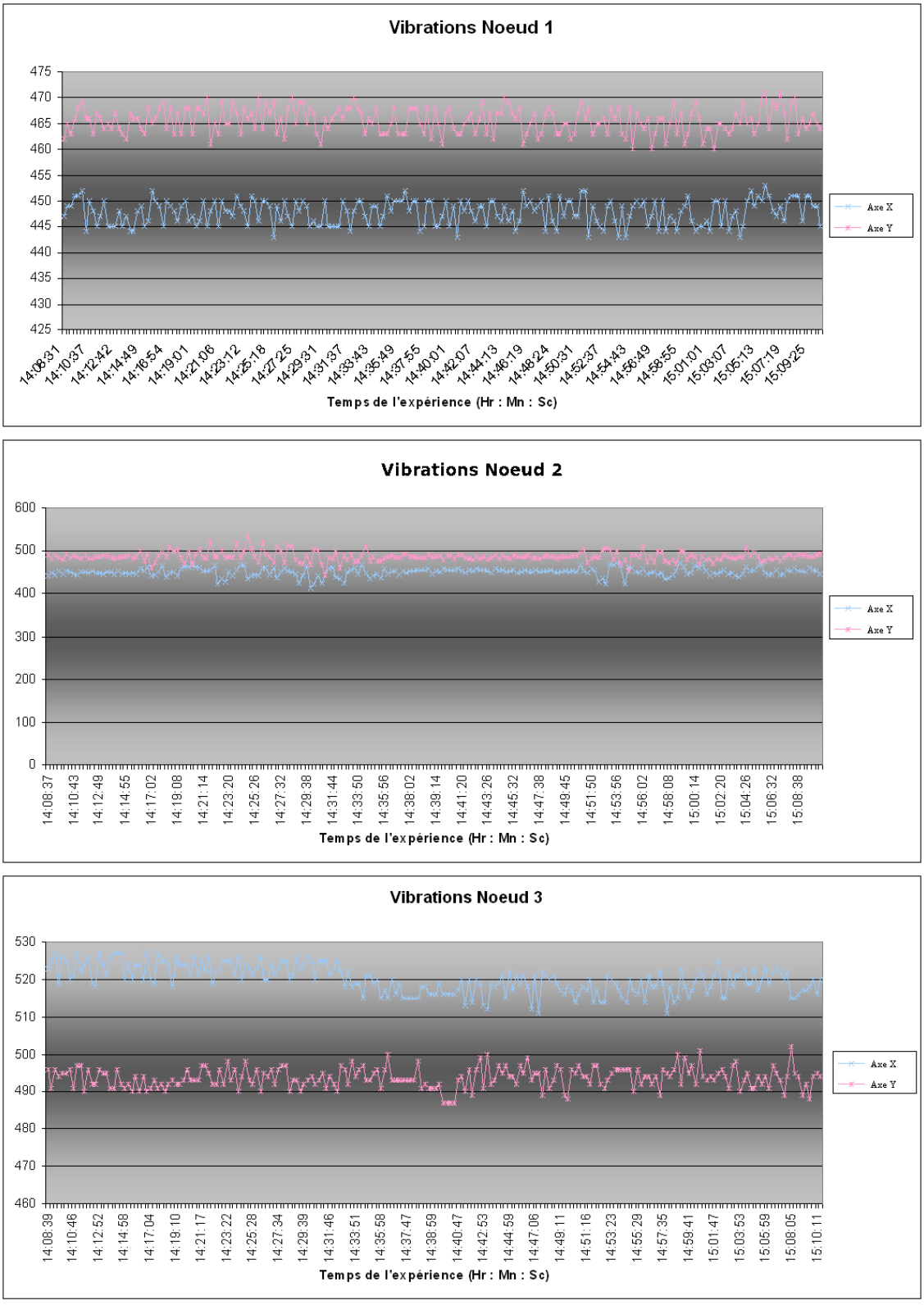


Figure 6.25*: Relevés de l'accéléromètre durant l'expérimentation*

Ces résultats, qui ne ceux sont pas réellement montrés significatifs durant notre expérimentation, nécessitent naturellement d'être complétés par d'autres tests.

Ce document a été rédigé en utilisant les logiciels LyX, LaTeX2e.

http://www.lyx.org/
http://www.tug.org/teTeX/

Vers une application des réseaux de capteurs sans fil dans la problématique des feux de forêts : Modélisation, Simulation et Plate-forme expérimentale

Résumé :

L'une des causes qui favorise les incendies de grande envergure est le manque de précison de l'alerte. En effet, les services de secours avouent avoir besoin d'outils capables de détecter et de préciser le siège d'un feu. Une nouvelle génération d'outil est apparue : les réseaux de capteurs sans fil. Dans une application de surveillance environnementale, ces réseaux de capteurs, possédant une architecture décentralisée et déployés dans une zone, doivent transmettre leurs informations, par une communication sans fil de proche en proche (ou multisauts), à une station de base. Pour cette communication multisauts, les techniques de routage s'attachent à réaliser une transmission efficace des données, tout en essayant d'économiser les ressources énergétiques. Dans le cadre de la problématique de lutte les feux de forêt, les réseaux de capteurs semblent capables d'aider les services de secours. Cependant, nous identifions trois axes de réléxion auxquels il faut fournir des réponses : ces réseaux sont-ils capables de prévoir, détecter et suivre un feu ? Pour répondre à ces questions, les outils de simulation existants s'avèrent incomplets ou inutilisables dans le cadre de notre étude. Dans cette dissertation, nous introduisons une nouvelle application, DEVS-WSN, basée sur une description à l'aide du formalisme DEVS. Elle permet d'étudier les performances et les stratégies de déploiement des réseaux de capteurs sans fil dans un feu. De plus, nous présentons un nouvel algorithme de routage VOX, qui améliore l'économie d'énergie et par conséquent augmente la durée de vie du réseau dans le contexte particulier de phénomènes destructeurs. Nous complétons cette étude théorique par une expérimentation en feu réel. Tous ces travaux avancent les conditions d'utilisation d'un réseau de capteurs sans fil dans la problématique des feux de forêt.

Towards an application of Wireless Sensor Network in Wildfire Fight : Modeling, Simulation and Testbed

Abstract :

One of the causes which supports large scale wildfires is the alarm precison lack. Indeed, firemen need tools able to detect fast and specify a sit of a fire. A new generation of tool is appeared : Wireless Sensor Network. In environmental monitoring, these sensors networks, having decentralized architecture and deployed in a zone, must to transmit their information, by a multi-hop communication, towards a base station. For this communication, the routing protocols attempt to carry out reliable data transmission, while testing to save the energy resources. In the problems of forest fires, the networks of sensors are able to help them firemen. However, we identify three axes for which it is necessary to provide answers : these networks, are they able to envisage, detect and to follow a fire ? To answer these questions, the existing simulation tools are incomplete or unusable in our context study. In this essay, we introduce a new application, DEVS-WSN, based on a DEVS description. It allows to study performance and the deployment strategies of the Wireless Sensor Networks. Moreover, we present a new algorithm of routing VOX, which improves the energy saving and increases lifetime of the network in the particular context of destroying phenomenons. We supplement this theoretical study by a test of a Wireless Sensor Netwok on real fire. All these work advances the conditions of use of Wireless Sensor Network in the fire problems.

Discipline, Spécialité : Sciences Physiques pour l'Environnement, Informatique
Mots clés : modélisation, simulation, évènements discrets, Réseaux de Capteurs sans fil, feux
Keywords : modeling, simulation, discrete events, Wireless Sensor Network, Wildfire

SPE (Systèmes Physiques pour l'Environnement), UMR CNRS 6134
Quartier Grossetti, BP 52, 20250 Corté
http://spe.univ-corse.fr