Supervision de réseaux d'objets intelligents communicants sans fil

Aurélien Jacquot

N° d'Ordre : 2019
EDSPIC : 477

Université Blaise Pascal - Clermont II

École Doctorale
Sciences Pour l'Ingénieur de Clermont-Ferrand

Thèse

présentée par

Aurélien Jacquot

Ingénieur en Informatique

pour obtenir le grade de

DOCTEUR D'UNIVERSITE

Spécialité : INFORMATIQUE

Supervision de réseaux d'objets intelligents communicants sans fil

---

Soutenue publiquement le 4 mars 2010 devant le jury :

Monsieur	Alain Quilliot	Président
Madame	Houda Labiod	Rapporteur
Monsieur	Bernard Tourancheau	Rapporteur
Madame	Sylvie Servigne	Examinateur
Monsieur	Jean-Pierre Chanet	
Monsieur	Kun Mean Hou	Directeur de thèse

A ma famille,
à Sandrine.

Remerciements

Je tiens tout d'abord à remercier les membres du jury d'avoir accepté de suivre cette thèse : Mme Houda Labiod et M. Bernard Tourancheau en tant que rapporteurs, Mme Sylvie Servigne en tant qu'examinatrice, M. Alain Quilliot en tant que président. Je les remercie pour le temps qu'ils m'ont accordé ainsi que pour les remarques et suggestions sur mes travaux.

Je remercie également mon directeur de thèse M. Kun Mean Hou pour son suivi et son aide tout au long de ces 3 dernières années, toujours là pour partager de nouvelles idées et concepts. Je souhaite remercier également M. Jean-Pierre Chanet qui au cours de ces années m'a encouragé et encadré au sein du Cemagref et sans qui je n'aurai sans doute pas pu faire cette thèse.

Je souhaite remercier le Cemagref de Clermont Ferrand et le conseil régional d'Auvergne pour leurs soutiens à la fois financier et matériel sans qui cette thèse n'aurait pu avoir lieu dans de si bonnes conditions.

Je remercie aussi les membres des différents comités de thèse, Mme Sylvie Servigne, M. Michel Schneider, M. Arnaud Lemarchand, M. André Miralles et M. François Pinet, pour leurs nombreux conseils qu'ils soient théoriques ou pratiques. Je remercie aussi l'ensemble de l'équipe COPAIN du Cemagref pour m'avoir fait une petite place dans cette grande famille. Plus globalement, je souhaite remercier M. Emmanuel Hugo ainsi que l'ensemble de l'UR TSCF, pour avoir facilité mon travail ainsi que les tâches administratives. Ensuite, au LIMOS, je remercie les membres de l'équipe SMIR pour leurs soutiens constants et leurs aides pour la réalisation et la validation des différentes plateformes matérielles développées. Je remercie plus particulièrement M. Frédéric Méniand pour sa disponibilité et son enthousiasme lors de la réalisation des différents prototypes plus ou moins fiables.

Enfin, une petite pensée à mes différents colocataires de bureau, dont Nicolas, Maud, Arnaud, et Oliver qui m'ont supporté pendant ces 3 années et permis de faire des pauses lors de journées bien chargées. Un petit clin d'oeil à Nicolas qui a commencé en même temps, avec qui j'ai pu vérifier mes bases en mécanique dont je garderai de très bons souvenirs et un très bon ami. Je remercie aussi les différents élèves avec qui j'ai pu travailler et qui m'ont permis d'avancer plus rapidement sur des points annexes.

Merci à vous tous et à celles et ceux dont je n'ai pas cité le nom mais qui m'ont permis d'avancer chaque jour un peu plus.

Pour finir les remerciements, je souhaite associer à cette thèse ma famille et mes amis qui m'ont permis de rester sur terre et garder une vie active bien remplie les weekends et autres vacances. Je remercie Sandrine (ma compagne), Angélique, Caroline, Sébastien, Valérie, Clément et les François, d'être présents jour après jour avec bonne humeur et des idées détente débordantes.

$\sim \star \sim \star \sim \star \sim$

Résumé

Les évolutions technologiques des dernières années ont permis de mettre en évidence un nouveau domaine de recherche, les Réseaux de Capteurs Sans Fil (RCSFs). Cette technologie est au 21ème siècle ce que l'ordinateur a été pour le 20ème. L'expansion de ces réseaux devrait être importante et occupera une large place dans notre quotidien. Avec l'utilisation de systèmes embarquées à faibles ressources, les RCSFs concentrent de nombreuses contraintes ainsi que de nombreuses problématiques de recherche. La gestion énergétique est l'un des enjeux majeurs de cette technologie. Développé à l'origine, pour la collecte de données environnementales, les RCSFs sont aussi utiles pour surveiller les infrastructures routières, les habitations intelligentes ou la santé, etc.

Une application type peut contenir plusieurs dizaines voire plusieurs milliers de noeuds capteurs. S'il existe déjà des méthodes pour diffuser, organiser et gérer les ressources, la problématique de la supervision du réseau reste entière. Les outils d'administration sans fil sont peu nombreux actuellement et ne répondent plus aux nouvelles contraintes des noeuds. Certaines d'entre elles s'appuient sur le protocole d'administration SNMP utilisé pour administrer les périphériques sur des réseaux filaires. Malheureusement, il est impossible d'implémenter ce protocole dans sa version classique du fait de l'espace mémoire restreint sur les noeuds. Cependant, SNMP est un protocole bien connu qui est utilisé sur beaucoup de réseaux par de nombreux périphériques. Avec sa notoriété, il semble intéressant de le prendre comme base pour administrer les réseaux d'OIC.

Nous proposons, dans ce manuscrit, une nouvelle solution d'administration nommée LiveNode Non invasive, Context aware, and Modular management (LiveNCM) pour répondre aux besoins d'administration d'un RCSF. Elle inclut des techniques innovantes, comme le diagnostic indirect ou les estimateurs, pour contrôler ses échanges et limiter son intrusion dans le fonctionnement de l'application embarquée. L'impact énergétique est alors plus faible que les solutions d'administration existantes. Avec une architecture hybride, les données centralisées sont directement accessibles par l'utilisateur depuis n'importe quel réseau interconnecté.

La mise en oeuvre de cette nouvelle approche de l'administration est illustrée par de nombreuses expérimentations en laboratoire mais aussi sur une application réelle telle que le suivi de l'humidité du sol en vu de piloter l'irrigation des parcelles agricoles. Avec LiveNCM, il devient alors possible de déployer des RCSFs sur de vastes zones rurales ou agricoles, tout en gardant un oeil distant sur chacun des noeuds.

Mots-Clés : Réseaux de Capteurs Sans Fil, Protocole SNMP, Diagnostic Indirect, Non invasif

Abstract

The technological developments of recent years have helped highlight a new research area, Wireless Sensors Networks. This technology is to the 21st century that the computer was to the 20th. The expansion of these networks should be important and will take a large place in our daily livings. With the usage of embedded systems with low ressources, WSNs concentrate many constraints and many research problems. Energy management is one of the major issues of this technology. Developed at the beginning for environmental data collection, WSN are also useful to manage road infrastructures, smart homes or smart care, etc.

A typical application can contain dozens or even thousands of sensors nodes. If there already are methods to communicate, organize and manage resources, the network management issue remains. Wireless Management Tools are few and no longer satisfy new constraints. Some of them are based on the SNMP Management Protocol used to manage devices on a wire network. Unfortunately, it is impossible to implement this protocol in its full classical version due to memory limitations on nodes. However, SNMP is a well known protocol which is used on many networks and devices. With this notoriety, it seems interesting to take it as a basis to manage WSN.

In this work, we propose, a new Wireless Management Tool named LiveNode Non invasive, Context aware and Modular management (LiveNCM) to meet the network management requirement. It includes new methods, such as indirect diagnosis or estimators, to control its exchanged data and to limit its intrusion (traffic overhead) into the application. The energy impact is lower than current methods ones. With hybrid architecture, centralized data are directly accessible by the user from any interconnected network with a standard SNMP tool.

The implementation of this new management approach is illustrated by many experiments in laboratory, but also on a real world application such as monitoring soil moisture for implementing intelligent irrigation system in precision agriculture. With LiveNCM, it becomes possible to deploy WSN on large rural or agricultural areas, and to monitor remotely each WSN node.

Keywords : Wireless Sensor Networks, SNMP protocol, indirect diagnosis, non invasive

Table des matières

Table des figures

Liste des tableaux

Liste des acronymes

6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
ADC	Analog to Digital Controller
AJAX	Asynchronous JavaScript and XML
ANMP	Ad-hoc Network Management Protocol
AODV	Ad hoc On-Demand Distance Vector
ASN.1	Abstract Syntax Notation 1
BOSS	Bridge Of the SensorS
CAN	Controller Area Network
CIVIC	Communication Inter Véhicules Intelligente et Coopérative
CMOS	Complementary Metal Oxide Semiconductor
CRC	Cyclic Redundancy Check
DANKAB	Directional Area Neighbour Adaptive Broadcast
DHCP	Dynamic Host Configuration Protocol
DTD	Document Type Definition
DYMO-low	Dynamic MANET On-demand for 6LoWPAN Routing
EENM	Execution Environment for Nomadic Manager
EEPROM	Electrically-Erasable Programmable Read-Only Memory
EQM	Erreur Quadratique Moyenne
FFCCC	Fairly Full Confidence Cooperative and Competitive
GMIB	Guerrilla Management Information Base
GSM	Global System for Mobile communication
GUERRILLA	Guerrilla management architecture
HiLow	Hierarchical Routing over 6LoWPAN
I^2C	Inter Integrated Circuit Bus
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
ISM	Industriel, Scientifique, et Médical
IP	Internet Protocol
LAN	Local Area Network
LEACH	Low Energy Adaptative Clustering Hierarchy
LIMOS	LIghtweight Multithreading Operating System
LIMOS	Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes
LiveFile	LIMOS Versatile Embedded File system
LOAD	6LoWPAN Ad Hoc On-Demand Distance Vector Routing

LiveNode	LIMOS Versatile Embedded Node
LiveNCM	LiveNode Non invasive Context-aware and Modular management
LiveNCM-P	LiveNCM communication Protocol
LoWPAN	Low-power Wireless Personal Area Network
LZW	Lempel-Ziv-Welch
MD5	Message Digest 5
MEMS	MicroElectroMechanical Systems
MIB	Management Information Base
OIC	Objets Intelligents Communicants sans fil
OLSR	Optimized Link State Routing Protocol
OS	Operating System
OSI	Open System Interconnection
PHP	PHP : Hypertext Preprocessor
QoS	Qualité de Service
RAM	Random Access Memory
RCSF	Réseaux de Capteurs Sans Fil
RFID	Radio Frequency IDentification
RLE	Run-Length Encoding
ROM	Read-Only Memory
RPL	IPv6 Routing Protocol for Low power and Lossy Networks
SensorML	OpenGIS Sensor Model Language Encoding Standard
SHAMAN	Spreadsheet based Hierarchical Architecture for MANagement
SNMP	Simple Network Management Protocol
SNMS	Sensor Network Management System
SPI	Serial Peripheral Interface
SPIN	Sensor Protocol for Information via Negotiation
SQL	Sructured Query Language
SSL	Spreadsheet Scripting Language
TCP	Transmission Control Protocol
TopDisc	Topology Discovery
TSCF	Technologies et Systèmes d'information pour les agrosystèmes
TTL	Time To Live
UDP	User Datagramme Protocol
UHF	Ultra High Frequency
UML	Unified Modeling Language
UWB	Ultra Wide Band
VLSI	Very-Large-Scale Integration
WAN	Wide Area Network
Wi-Fi	Wireless Fidelity
WSN	Wireless Sensor Networks
WSG	Wireless Sensor Gateway
XML	eXtensible Markup Language

$\sim \star \sim \star \sim \star \sim$

Introduction

Depuis ces 50 dernières années, l'informatique et l'électronique connaissent une expansion constante modifiant ainsi nos modes de vies. Depuis le premier ordinateur qui était imposant, lourd et peu performant, la miniaturisation et le gain de performances ont été les fils directeurs de tout développement matériel et logiciel dans le domaine des systèmes embarqués. Si l'ordinateur a été l'un des bouleversements les plus importants de notre quotidien, les Réseaux de Capteurs Sans Fil (RCSFs) ou Wireless Sensor Networks (WSN) seront sans aucun doute la technologie clé du 21ème siècle. La miniaturisation toujours plus poussée d'année en année permet aujourd'hui de fournir des plateformes matérielles aussi puissantes que des ordinateurs du début des années 90. Le but de ce nouveau domaine technologique est de répondre à des applications de collecte de données ou en remplacement d'infrastructures existantes par exemple. Dans le contexte des RCSFs, ce sont les performances énergétiques qui sont les plus importantes, par rapport aux performances matérielles (processeur, mémoire). L'augmentation de l'une se fera nécessairement au détriment de l'autre. Une gestion plus intelligente des ressources mémoire ou énergétique impliquera une augmentation de l'utilisation de la ressource processeur et/ou mémoire. Par ailleurs, l'explosion des communications sans fil permet aujourd'hui de disposer de médiums de communication autorisant une couverture de vastes zones. Les RCSFs peuvent être assimilé à nos ordinateurs du passé miniaturisés et combinés aux nouvelles technologies sans fil.

Contexte de l'étude

Le domaine applicatif des RCSFs n'a de limite que les capteurs permettant d'acquérir l'information. Qu'ils soient utiles pour les militaires, l'agriculture, les infrastructures routière, la santé, ou les maisons intelligentes, les RCSFs constituent de nos jours, un des domaines de recherche les plus actifs. L'évolution prédite pour eux est à l'image de l'ordinateur dans nos foyers. Le futur verra fleurir des noeuds sans fil reliés en réseau pour suivre, piloter ou contrôler les conditions d'évolution des personnes, des ressources naturelles, ou des habitations. Au sein du l'unité de recherche Technologies et Systèmes d'information pour les agrosystèmes (TSCF) du Cemagref et du Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes (LIMOS) UMR 6158 CNRS, la problématique des réseaux agri-environnementaux a été choisi pour faciliter la collecte des informations dans des milieux ruraux et difficiles d'accès. Le domaine de prédilection est donc l'agriculture et la surveillance environnementale avec toutes les problématiques écologiques qui peuvent y être associées.

Ce domaine applicatif présente de nombreuses problématiques et contraintes tant au niveau matériel que logiciel. L'utilisation de noeuds à ressources limités engendre de nouvelles problématiques comme l'optimisation des applications tant au niveau temps d'exécution qu'au niveau de l'espace mémoire, et amplifie les contraintes existantes comme la communication sans fil, et la consommation énergétique. Les différents acteurs du domaine, au cours de ces dernières années, ont proposé des protocoles de routage, de mécanisme de tolérance aux fautes, ou des méthodes de développement modulaire plus ou moins efficace en fonction du type d'applications. Cependant, un point a été moins exploré : l'administration de ces réseaux. Si dans un premier temps cette problématique n'a pas eu beaucoup d'importance, il est nécessaire aujourd'hui de l'étudier en détail pour disposer d'un RCSF viable et fonctionnel.

Objectifs et contributions de la thèse

Les RCSFs constituent un domaine de recherche très attractif tant pour les électroniciens, les informaticiens que pour les mathématiciens. La mise en place de tels réseaux demande de disposer d'un ensemble de méthodes et de techniques pour diffuser des données, les collecter et les traiter. La gestion de chaque noeud demande d'avoir des solutions efficaces permettant un accès simple et rapide aux données des noeuds. La contribution de cette thèse est de proposer une réponse à cette problématique afin de rendre utilisable un RCSF en milieu agricole et rural. L'objectif majeur de la solution d'administration LiveNCM est centré sur la préservation des ressources énergétiques et matérielles afin de limiter son impact sur l'application et sur le réseau. Par définition, elle minimise ses échanges en prenant en compte le contexte de mesure et l'ensemble des messages pour déduire des informations d'administration.

Pour cela, de nouvelles techniques comme le diagnostic indirect devront être mise en place pour favoriser l'administration en minimisant les échanges. En complément, des mécanismes de réduction du volume de données pourront être utilisés comme les estimateurs ou la compression. Ensuite, une nouvelle architecture d'administration sera développée pour répondre au besoin d'accès aux données. Dans un réseau filaire classique, l'accès à chaque machine est simple et rapide sans vraiment de contraintes de communication ou de routage. Chaque noeud peut embarquer un agent ou une brique logicielle pour répondre au besoin d'administration. Dans les RCSFs, les noeuds pouvant être hors de portée du superviseur, un ensemble de méthodes d'organisation et de routage devront être mise en place. De plus, l'accessibilité des noeuds peut faire défaut à tout instant, c'est pourquoi il est important pour l'utilisateur de voir les problèmes et autres erreurs survenues sur un noeud. A chaque information souhaitée, les réponses courantes d'administration consistent à former une requête et à la diffuser sur le réseau. Il est facile de constater que dans ce cas, certains noeuds relais et plus globalement le réseau, auront une autonomie diminuée. LiveNCM propose de centraliser les données afin de limiter le nombre de requêtes sur les noeuds, mais aussi afin d'améliorer l'accessibilité des informations aux différents utilisateurs locaux ou distants.

Cette thèse essaye de répondre à la problématique de supervision de RCSFs, en proposant une nouvelle solution d'administration LiveNCM, qui prend en compte un ensemble important de contraintes subies par les noeuds. La contrainte énergétique influencera tous les développe-

ments autour de LiveNCM, et la minimisation de l'impact de l'administration sur les noeuds constituera une nouvelle approche pour augmenter la durée de vie d'un RCSF.

Organisation du mémoire

Ce mémoire est constitué de quatre chapitres. Le premier est consacré à la présentation du contexte des RCSFs, leurs applications et leurs contraintes majeures. Nous verrons les différents éléments présents dans ces réseaux, avec une présentation plus détaillée d'un noeud et des fonctionnalités qu'il peut embarquer. Le second chapitre présente l'état de l'art des techniques d'administration dans les réseaux à la fois filaires et sans fil. Dans cette partie, nous présenterons aussi les méthodes annexes qui permettent d'améliorer la diffusion des données et la préservation de l'énergie sur les noeuds. Qu'elles soient basées sur un protocole existant ou sur une architecture spécifique, nous étudierons les différentes solutions pour en tirer les conséquences et poser les premières pierres pour une nouvelle solution d'administration. Le troisième chapitre présente la solution LiveNCM en détails. Une première partie présentera les techniques mises en oeuvre pour réduire les échanges dans les réseaux et sur les noeuds afin de garantir un impact très faible sur le fonctionnement général de l'application. Ensuite les parties suivantes s'attarderont sur les parties logicielles constituant l'architecture de la solution LiveNCM. Le quatrième chapitre évaluera notre contribution sur un réseau agri-environnemental. Une évaluation des différentes parties, l'agent SNMP, la passerelle de communication et l'agent embarqué sur les noeuds, sera présentée. Dans ce chapitre, nous verrons l'impact de l'estimation sur l'autonomie des noeuds et sur les volumes de données engendrés. Enfin, en fin de chapitre nous présenterons une application réelle ayant pour but le suivi de l'humidité du sol en vue de piloter les systèmes d'irrigation sur une parcelle agricole. Pour finir, une conclusion synthétisera les différents points abordés dans ce mémoire, avec un bilan de notre contribution. Finalement, des perspectives de développement seront présentées pour fixer les futures évolutions de notre solution d'administration LiveNCM.

$\sim \star \sim \star \sim \star \sim$

Chapitre 1

Les réseaux de capteurs sans fil

1.1 Origine et développement

Les différentes évolutions technologiques de ces dernières années, à la fois dans le domaine des capteurs MicroElectroMechanical Systems (MEMS), des circuits intégrés Very-Large-Scale Integration (VLSI) et des réseaux sans fil, ont permis le développement d'un nouveau domaine de recherche : les Réseaux de Capteurs Sans Fil (RCSFs) ou Wireless Sensor Networks (WSN). A l'origine militaire, cette technologie constitue aujourd'hui un sujet de recherche en pleine expansion, car elle est une des technologies clé du 21ème siècle et ses applications innombrables. De nombreuses problématiques telles que la durée de vie, la fiabilité et l'encombrement, constituent le coeur de ce domaine, ce qui en fait un important sujet de recherche actuel et pour le futur dans le cadre du développement durable de nos sociétés [Chong 03, Puccinelli 05].

1.1.1 Quelques exemples d'applications

Le domaine d'application de cette technologie n'a de limite que les bornes fixées par les grandeurs physiques observables. Les premières applications étaient principalement militaires et orientées autour de la détection d'intrusion sur un champ de bataille [Akyildiz 02]. Pour cela, la zone à surveiller était préalablement équipée de capteurs munis de détecteurs de présence (capteurs ultrason, capteurs de pression, etc.) et disposant d'un médium de communication à longue portée (ondes radio Ultra High Frequency (UHF)).

Sur la figure 1.1, la détection d'une intrusion, infanterie ou véhicules, se traduit par une alerte vers le camp de base décrivant la nature de l'intrusion, la localisation, et la taille de l'intrusion. Le RCSF s'adapte alors en créant un périmètre autour des intrus pour en suivre l'évolution. Par ailleurs, à l'aide des informations fournies, la riposte pourra être effectuée de manière optimale sans risquer des pertes [Bokareva 06]. Cette application bien que basique demande de garantir les échanges d'informations entre les noeuds et la station de base, ici le camp de base. Le futur des applications militaires se tourne vers l'appui aux soldats en les équipant de dispositifs permettant de suivre leur santé, l'état de leurs réserves alimentaires et de leurs munitions, et de les aider à évoluer sur un terrain inconnu. Ces dispositifs seront

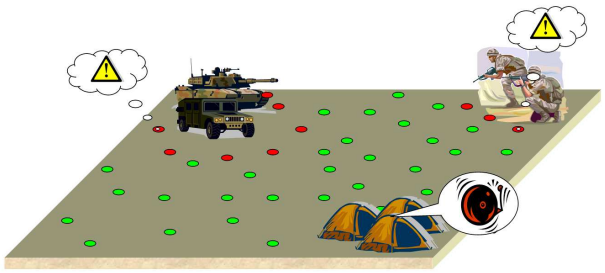

Figure 1.1 – Détection d'intrusion dans une zone militaire surveillée

aussi déployés sur les différents véhicules pour organiser les ravitaillements et les éventuelles réparations avant même l'arrivée du véhicule à son garage.

A partir de ces applications, il est apparu intéressant de développer cette technologie sur des applications civiles afin de mieux comprendre et interagir avec notre environnement [Wang 06]. Les politiques écologiques actuelles ont permis d'utiliser les RCSFs pour observer des phénomènes environnementaux et plus particulièrement ceux concernant les activités agri-environnementales [Akyildiz 06, Jacquot 07a]. La différence entre ces deux applications réside dans le type de grandeur physique observée et le type de traitement associés. Le premier permet essentiellement de se prémunir et de prévenir les personnes d'un danger potentiel (pollution de l'air ou des eaux, gaz toxiques, etc.), tandis que le second se concentre sur les aspects plus techniques de la gestion des ressources naturelles (niveau d'eau, pollution engendré par l'agriculture, etc.). Ainsi, il est courant de trouver des systèmes d'aide à l'irrigation de domaines viticoles [Matese 09], de domaines de loisirs (Figure 1.2), ou de parcelles agricoles [Net-Added 09].

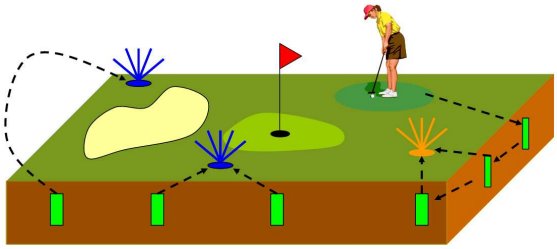

Figure 1.2 – Aide à l'irrigation de centre de loisirs

Sur la figure 1.2, le RCSF est enterré et détecte les variations d'humidité pour pouvoir contrôler l'arrosage du parcours de golf. Ces actions peuvent se faire automatiquement ou manuellement par le jardinier ou par un opérateur distant qui traitera les informations reçues. Il est évident que pour un fonctionnement automatique, le RCSF sera capable de détecter les personnes via un tag Radio Frequency IDentification (RFID) par exemple. Dans ce contexte, les RCSFs récoltent des données pour mieux comprendre son évolution et pour agir au mieux. Les applications d'aide à l'irrigation constituent un enjeu économique et social très important face aux éventuelles pénuries d'eau annoncées dans les années à venir. C'est entre autres pour

cela qu'une très grande majorité des RCSFs servent aux cultures (céréalières, viticoles, maraîchères) et à l'aide à la compréhension de notre environnement. [Wang 06, López Riquelme 09, Beckwith 04]. Par ailleurs, les catastrophes naturelles comme les éruptions volcaniques ou les inondations constituent des applications importantes [Werner-Allen 05]. En dehors de l'aspect récolte de données, la sécurité et la fiabilité des échanges sont au coeur du développement des ces applications ; le but étant d'alimenter une base d'informations afin d'alerter au plus tôt les personnes, les informations reçues pourront être traitées automatiquement dans le cadre d'un plan d'alerte automatisé par exemple. Suivant les observations, il est tout à fait envisageable de prévenir par le biais de messages les personnes concernées par l'alerte via leur téléphone portable, leur email ou les moyens traditionnels de diffusion de l'information (télévision, radio).

Notre quotidien va aussi connaître l'explosion des RCSFs, pour aider à fluidifier la circulation [Coleri 04], à contrôler la pollution [Brunet 01], à lutter contre les dangers qui nous entourent comme le feu [Chaczko 05] ou alors tout simplement à nous aider dans nos tâches du quotidien [Mainwaring 02]. Ainsi, dans un futur proche, chaque véhicule pourra échanger des informations avec les usagers mais aussi recevoir des alertes sur un accident ou autres perturbations de la circulation. Les infrastructures routières intelligentes auront la possibilité de diffuser des informations avec des panneaux d'affichage et de s'organiser automatiquement pour prévoir des plans de déviation. Une évolution de cette application pourrait permettre de contrôler les flux routiers en un point d'une ville pour abaisser les taux de pollution. Enfin, au quotidien, un RCSF utilisé sur un réseau de transports en commun peut aider à la prise en charge des personnes à mobilité réduite ou des usagers en fauteuil roulant [Zhou 06]. Les chauffeurs seront alertés de la présence d'usagers à prendre en charge sans même avoir vu l'arrêt de bus. Il adaptera alors son approche pour permettre une accessibilité optimale de l'usager à l'autobus. Par ailleurs, les RCSFs ne se résument pas à aider les déplacements des personnes. On retrouve aussi cette technologie dans le domaine de la santé pour suivre un patient à distance, pour prévenir certains malaises cardiaques [Zhou 04, Chelius 09] par exemple. Le patient n'est plus contraint de rester à l'hôpital et en cas de problème le médecin est directement averti via un email ou un message sur son téléphone portable par exemple.

1.1.2 Définition, caractéristiques et objectifs de cette technologie

Par définition, les RCSFs sont constitués d'un ensemble de noeuds disposant de capteurs, ou d'Objets Intelligents Communicants sans fil (OIC), échangeant des informations entre eux à l'aide d'un medium d'accès avec ou sans fil [Akyildiz 02]. Le noeud est défini comme la partie disposant de l'intelligence et de la communication tandis que le capteur représente la partie réalisant la mesure physique.

Chacun d'eux dispose d'un moyen de communication utilisant un médium de communication et d'une partie intelligente exécutant le système embarqué et les applications, pour interfacer les différents capteurs employés (Figure 1.3). Les noeuds peuvent être nommés soit "noeuds évolués" [Hou 07a, Hill 01, Dubois-Ferriere 06], pour ceux de taille réduite, soit "noeuds poussière" (smart dust), [Kahn 99] pour des tailles très faibles (dimension d'un grain de riz voir plus petit). La différence entre ces deux types est principalement liée aux fonctionnalités em-

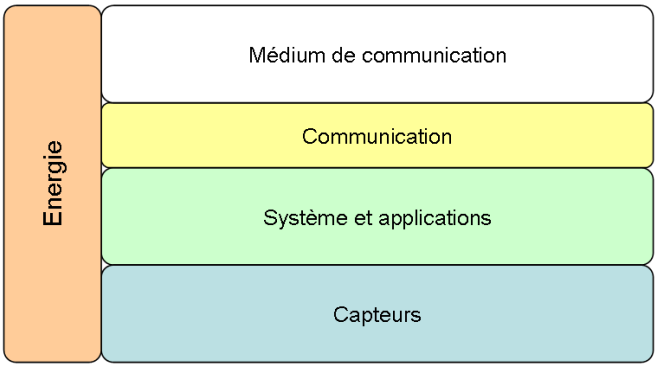

Figure 1.3 – Différents éléments d'un noeud

barquées, au nombre de capteurs employés et à leur autonomie. Dans le premier cas, on utilisera quelques dizaines de noeuds pour couvrir une zone à observer, tandis que dans le second cas, on en sèmera plusieurs milliers. Bien souvent, ils n'embarquent qu'un ou deux capteurs pour réaliser une tâche bien spécifique. Les coûts de ce type de capteurs sont relativement faibles [García-Hernando 08] grâce à l'utilisation de matériaux courants et simples à mettre en oeuvre ce qui n'autorise, dans la majorité des cas, qu'un usage unique des capteurs, tandis que les noeuds évolués permettent une certaine modularité en embarquant plus de ressources afin de pouvoir gérer plusieurs capteurs externes et exécuter des prétraitements directement sur le noeud. Les applications sont donc plus variées et peuvent être basées sur des capteurs simples de température, de pression ou d'humidité ou alors sur des capteurs plus évolués comme les capteurs de pollution (air, eau), ou de radioactivité. Pour ces noeuds, le coût unitaire est certes plus élevé mais il reste globalement faible puisqu'il y en aura moins à déployer pour effectuer la même fonction et ils pourront être réutilisés pour différentes applications sur différents lieux.

Sur les noeuds de type poussière, la complexité des applications étant très faible, il est préférable d'utiliser des noeuds évolués pour mettre en place les différentes briques logicielles et matérielles d'un RCSF. Ils permettront de concevoir et tester la logique embarquée comme le système d'exploitation et les méthodes de transmission avec les protocoles de routage. Par ailleurs, il sera possible d'évaluer ces performances en vue d'une intégration future à taille réduite. C'est pourquoi actuellement, sur les applications de type agri-environnementales, on retrouve ce raisonnement avec des noeuds capteurs plutôt imposants pour évaluer leur efficacité tout en gardant à l'esprit une future réduction de leur taille et de leur coût.

Par ailleurs, les échanges dans ces réseaux constituent encore aujourd'hui un incroyable défi technologique afin d'allier portée et débit tout en réduisant le coût énergétique d'une transmission. Les premiers modules ont embarqué des sources de transmission radio utilisant les bandes de fréquences Industriel, Scientifique, et Médical (ISM) libres (433MHz ou 915MHz). Puis avec le développement de nouvelles technologies de communication, les noeuds sont devenus de plus en plus petits et de plus en plus performants en termes de communication. Les normes les plus répandues sont maintenant IEEE 802.11 et IEEE 802.15 qui utilisent des bandes de fréquence ISM plus élevées (2.4 GHz et 5 GHz). La norme Wireless Fidelity (Wi-Fi), IEEE 802.11

[IEEE 07], est déclinée en différentes versions qui apportent une amélioration tant sur la qualité de service que sur les débits proposés. Cette norme se décline sous 13 versions différentes pour des applications domestiques (réseau sans fil individuel IEEE 802.11.(b/g/n)) ou industriel (IEEE 802.11.(a/b)). Son intérêt réside dans les débits proposés qui prévoient un débit théorique allant de 11Mb/s (IEEE 802.11.b) jusqu'à 600 Mb/s (IEEE 802.11n [IEEE 09]). La contrepartie de ces débits est l'augmentation de la consommation énergétique des modules de communication. C'est pourquoi, les plateformes actuelles ne l'utilisent que très peu. En réponse à cette problématique, il a été créé la norme IEEE 802.15 regroupant à la fois la technologie Bluetooth (IEEE 802.15.1) [IEEE 05] et ZigBee (IEEE 802.15.4) [IEEE 03]. Le Bluetooth, qui est maintenant sur tous les appareils mobiles domestiques de notre quotidien (téléphone portable, modem ADSL, télévision...), alterne périodiquement les fréquences pour vaincre les bruits et obstacles, et proposer un débit maximal théorique de 1 Mb/s pour la version 1.0 et de 3Mb/s pour la version 2.0. La faible portée de cette technologie et les restrictions du concept maître esclave (1 maître et 7 esclaves) l'ont éloigné du domaine des RCSFs pour être plutôt utilisée sur des applications domestiques de téléphonie et de transfert de données. A la différence du Bluetooth, la technologie ZigBee est très répandue car elle a été créée afin d'avoir de grandes portées (jusqu'à plusieurs kilomètres) pour une faible consommation (quelques milliwatts). L'inconvénient du ZigBee est son faible débit (256 kb/s) mais dans la majorité des applications, n'est que très peu pénalisant car les données échangées sont constituées de messages de petite taille. Dans le futur, les différents acteurs du secteur des RCSFs s'orientent vers un nouveau médium de communication sans fil ultra large bande, Ultra Wide Band (UWB) [Ellis 02], qui doit permettre de réduire la consommation énergétique, tout en améliorant le couple portée et débit. Ce compromis est en partie dû à la simplicité de l'architecture matérielle employée pour la modulation et la transmission des ondes. Cependant, les difficultés éprouvées pour définir ce standard bloquent son évolution et par conséquent sa diffusion. La figure 1.4 compare les différents médiums de communication employés dans les RCSFs.

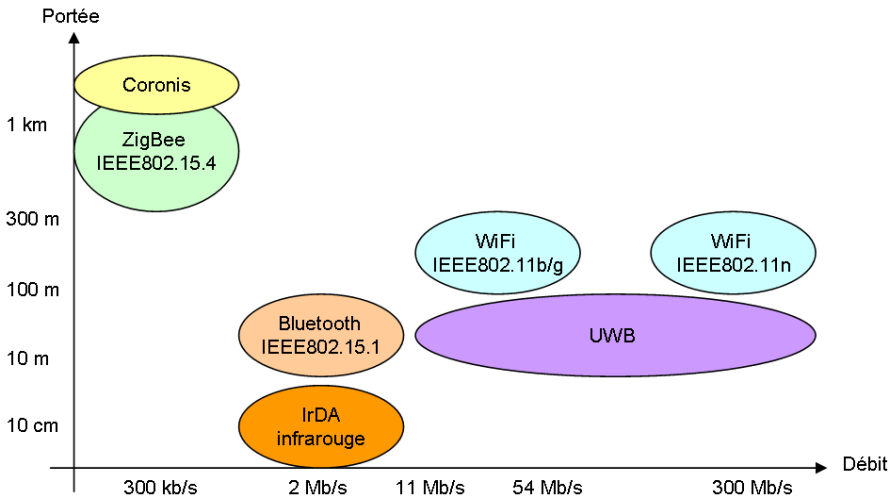

Figure 1.4 – Comparatif des différentes normes de communication sans fil

Table 1.1 – Comparatif des plateformes existantes

	LiveNode (2007) [Hou 07b]	TinyNode (2006) [Dubois-Ferriere 06]	Mica2 (2001) [Hill 01]
Processeur	Atmel AT91SAM7S256 RISC 32bits	Texas Instruments MSP430 RISC 16bits	Atmel ATmega128L RISC 8bits
Fréquence	500Hz à 55MHz	8MHz	jusqu'à 16MHz
Mémoire Flash	256kB	64kB	48kB
Mémoire RAM	64kB	10kB	10kB
Médium sans fil	WiFi | ZigBee IEEE 802.11b | IEEE 802.15.4	ZigBee IEEE 802.15.4	868/916 MHz
Portée (m)	50/200 | 300/1600	40/200	40/150
Débit théorique	11 Mb/s | 250kb/s	1.2-152.3 kb/s	38.4 kb/s

Il est à noter que les normes, Bluetooth (IEEE 802.15.1) et ZigBee (IEEE 802.15.4) sont plus adaptées aux applications à faible consommation énergétique où elles ne nécessitent pas de débits importants, tandis que UWB (IEEE 802.15.3a) et Wi-Fi (IEEE 802.11) sont plus utiles pour des applications à fort débit telles que la diffusion de l'audio et de la vidéo [Jin-Shyan 07]. Il est par contre intéressant de constater que les études de [Jin-Shyan 07] montrent que le rapport entre la consommation énergétique et les débits est très mauvais pour les technologies Bluetooth et ZigBee ; de 150 à 300 mJ/Mb pour ces derniers contre 6 à 13 mJ/Mb pour les autres. Pour finir, il est donc intéressant d'utiliser des médiums de type ZigBee pour leurs consommations énergétiques en contrôlant leur durée d'utilisation.

Cependant, l'architecture des noeuds ne se résume pas seulement au moyen de communication. Ils embarquent chacun une unité de traitement, un processeur à faible consommation, et une unité de stockage, souvent intégrée au processeur. Le fil rouge des RCSFs étant la réduction de la consommation énergétique, il a été nécessaire de trouver un compromis entre la puissance de calcul et le coût en ressources. Ainsi, on retrouve sur de nombreuses plateformes des processeurs basse consommation du type ATMEL ATMega [ATMEL 09], ATMEL ARM7/9 [ATMEL 05], ou Texas Instruments MSP [Texas-Instruments 09] qui peuvent fonctionner à de faibles fréquences tout en ne consommant que quelques milliwatts. Le tableau 1.1 présente brièvement les caractéristiques de noeuds présents sur le marché.

Avec le temps, de droite à gauche dans ce tableau, les performances des noeuds ont été significativement augmentées et les capacités mémoire ont été multipliées au moins par 4. Pour les applications utilisant des noeuds évolués, cela se traduit par le déploiement de prétraitements de plus en plus complexes directement sur les noeuds, mais aussi par l'intégration d'algorithme plus performant pour organiser les transferts de données dans les RCSFs. Le coût énergétique des prétraitements embarqués reste négligeable comparé au coût d'une transmission.

Pour finir, l'élément le plus important dans tout système électronique ou mécanique est

la source d'énergie. Les noeuds utilisent des sources d'énergie électrique variées pouvant être de simples piles du commerce jusqu'aux sources d'énergie renouvelable telles que l'éolien ou le solaire [Roundy 04]. Cependant, les solutions d'alimentation se réduisent considérablement dès que l'application nécessite que les noeuds soient déployés dans des endroits confinés ou sous terre. Dans ces cas, les piles rechargeables ou non restent pour le moment la seule solution efficace. Dans le futur, les piles pourront être remplacées par des dispositifs plus performantes comme des piles à combustible (hydrogène par exemple) si leur taille venait à réduire suffisamment pour permettre une intégration facile avec un faible encombrement et un bon rendement. On a donc besoin à ce jour d'optimiser l'utilisation de cette ressource non renouvelable afin d'atteindre les objectifs d'autonomie et de fonctionnement désirés. Par contre, en extérieur, les sources d'énergie renouvelable telles que le solaire [Sun 09], l'éolien ou l'hydraulique [Morais 08], sont répandues ce qui garantit un fonctionnement sur de longues durées éventuellement infinies, la seule limite restant la durée de vie d'une batterie. Cependant, de telles sources intarissables ne permettent pas d'ignorer les contraintes énergétiques d'un noeud lors des nuits ou des périodes sans vent. La tendance étant à la miniaturisation, la consommation devra être réduite afin de limiter au maximum la taille du panneau solaire ou de l'éolienne utilisée. En effet, les panneaux solaires de faible taille aujourd'hui n'ont qu'un rendement très faible (15% pour les meilleurs [IEA 09]) ce qui oblige de contrôler finement l'utilisation de cette ressource afin de permettre un rechargement des batteries lors des phases d'ensoleillement. Pour des applications utilisant une mini centrale hydraulique, ce problème énergétique devient moins important car la source d'énergie est quasiment inépuisable et reste présente à tout moment de la journée. Enfin, la dimension à prendre en compte dans les RCSFs est l'impact environnemental de cette technologie. En effet, l'utilisation de piles améliore certes l'autonomie mais produit aussi des déchets nuisibles. Cela pose le problème de savoir comment récupérer les noeuds usagés ou comment produire des noeuds efficaces énergétiquement et si possible utilisant des énergies renouvelables.

1.2 Contraintes subies par les noeuds

La diffusion des RCSFs sur des applications domestiques ou sensibles a demandé de prendre en compte de nombreuses contraintes variées et bien souvent liées. Les applications présentées montrent à quel point, les contraintes subies par cette technologie peuvent être variées et difficiles à contrôler. Un premier classement de ces contraintes peut être proposé pour les regrouper :

- *Matérielle* : Faible source d'énergie embarquée, consommation énergétique ;
- *Système* : Faible ressources processeur et mémoire, traitements limités ;
- *Communication* : Portée, bande passante du médium de communication, Qualité de Service (QoS), collisions, et interférences ;
- *Environnement* : Tolérance aux pannes, conditions d'évolution, agressions externes ;
- *Capteurs* : Calibration, coefficients de mesure, qualité des mesures.

Pour illustrer leurs interactions, la figure 1.5 représente les différentes contraintes ainsi que les liaisons entre elles. On devine facilement que l'un des points communs à toutes ces contraintes est lié à la source énergétique disponible ce qui en fait un problème récurant pour tout futur développement d'applications.

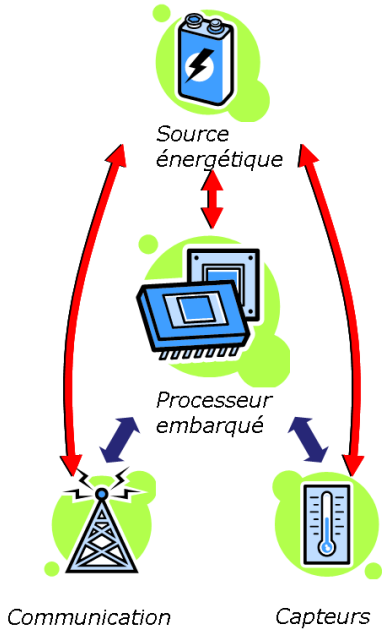

Figure 1.5 – Influence de l'énergie sur les contraintes d'un noeud

1.2.1 Contraintes 'Matérielle'

La contrainte '*Matérielle*' est majeure sur les plateformes matérielles et sur les applications embarquées ; elle se concentre principalement sur la ressource énergétique. Les noeuds ne disposant que d'une faible source d'énergie, il est absolument indispensable d'optimiser leur utilisation. Pour cela, il est possible d'agir sur différents points pour réduire la consommation énergétique. Tout d'abord, il est important de développer un noeud qui consomme peu d'énergie en utilisant des technologies basse consommation CMOS (Low Power CMOS). L'objectif aujourd'hui est de produire des plateformes ne consommant que quelques micro-ampères [Eriksson 08]. Pour cela, une telle conception demande une optimisation du fonctionnement de l'application tant sur les traitements que sur l'utilisation des différents composants. Ainsi, il doit être possible de moduler la fréquence de fonctionnement du processeur ou d'éteindre certaines parties inutiles du noeud depuis celui-ci ou à distance. L'ajout d'un module autonome de contrôle de l'énergie peut constituer un avantage non négligeable concernant l'autonomie des noeuds. Son objectif premier sera de contrôler l'activation des différents périphériques disponibles ou de moduler les fréquences de chacun en détectant les périodes de faibles activités. Une autre solution consiste à informer l'utilisateur de l'état du noeud et à lui laisser la possibilité d'agir à distance via un module d'administration. Cela peut se traduire par un outil

d'administration simple, pour envoyer des traitements à effectuer, ou évolué, pour agir sur les différents paramètres du noeud. En complément, la finalité des RCSFs étant d'être diffusée très largement, la taille de la plateforme matérielle entre en compte dans sa conception. Un système miniaturisé ne consommera que peu d'énergie ce qui est un point fort, mais la contre partie est de limiter les performances du système et sa modularité.

1.2.2 Contraintes 'Système'

Les contraintes '*Système*' regroupent celles liées aux *ressources processeur et mémoire du noeud*. En effet, l'unité de stockage (mémoire interne ou externe) d'un noeud est souvent limitée à la mémoire incluse avec le microcontrôleur. L'unité de stockage regroupe les mémoires internes Random Access Memory (RAM), Read-Only Memory (ROM), et Flash de type Electrically-Erasable Programmable Read-Only Memory (EEPROM) ainsi que les modules externes de mémoire Flash de type EEPROM par exemple. Il est donc nécessaire de porter une attention particulière sur son utilisation. Pour cela, l'empreinte mémoire d'une application doit être optimisée afin de laisser un maximum d'espace aux données utilisateur mais aussi pour permettre leurs mises à jour à distance. Il est toujours possible d'étendre cet espace de stockage en ajoutant un module de gestion de mémoires externes. Les inconvénients de cette vision sont d'ajouter un module consommateur d'énergie et de complexifier l'accès aux données. En outre, la détection des fuites de mémoire ainsi que des outils efficaces de gestion de la mémoire sont nécessaires pour répondre à cette contrainte. Différents travaux comme [De Sousa 08] portent sur le développement de système de fichiers embarqué et utilisant l'ensemble des noeuds comme un immense lieu de stockage. En effet, quelques systèmes de fichiers comme LIMOS Versatile Embedded File system (LiveFile) [De Sousa 07] implémentent des méthodes de distribution de données basées sur un stockage déporté sur différents noeuds d'un même groupe par exemple.

De plus, les caractéristiques des processeurs ou unités de calcul généralement utilisées sur les noeuds (ATMEL ATMega [ATMEL 09], Texas Instruments MSP [Texas-Instruments 09] ou ATMEL ARM7/9 [ATMEL 05]) ne permettent pas de réaliser des calculs longs et complexes du fait de leur architecture, de leurs fréquences de fonctionnement et bien sur de leurs mémoires internes. C'est pourquoi, il est courant d'intégrer un noeud plus performant dans les réseaux pour soutenir des calculs plus complexes d'organisation ou de traitements des données, la station de base ou 'sink node'. Sur ce point, ces noeuds doivent pouvoir voir l'ensemble du réseau ou disposer de méthodes d'accès aux différents éléments. Par ailleurs, la consommation énergétique de ces microcontrôleurs est en lien direct avec la fréquence de fonctionnement et les périphériques utilisés. A l'aide d'une modulation de la fréquence du processeur, des économies sont effectuées mais par contre les temps de traitements sont plus longs. On retrouve ce mode de fonctionnement sur les ordinateurs portables qui ralentissent le processeur lors des phases d'inactivité ou de faible sollicitation. Ensuite, une bonne gestion des périphériques allumés sur le processeur offre une meilleure durée de vie au noeud. Ici aussi, le modèle est comme les ordinateurs portables qui en mode batterie choisissent d'éteindre des périphériques externes comme les interfaces réseau par exemple.

1.2.3 Contraintes 'Communication'

Les contraintes de '*Communication*' regroupent toutes les perturbations que peut subir une transmission de données d'un point A à un point B. Dans ce contexte, il est donc nécessaire de faire attention au taux de pertes des données afin de limiter le nombre de retransmissions. Pour cela, une attention particulière doit être portée au choix du médium de communication sans fil ainsi qu'au choix de son antenne et de son emplacement. Les portées des médiums de communication sans fil sont directement associées aux conditions d'utilisation et à l'environnement. La dispersion des ondes dans un milieu peut être calculée dans un cas idéal mais en pratique, les échanges ne répondent pas aux mêmes règles à cause des perturbations externes ou des infrastructures proches. En agissant sur la portée ou sur la puissance d'émission, il est possible de garantir une bonne réception des informations pour les noeuds les plus proches, ceux en limite de portée seront sujets à un taux de pertes importants du fait de la faible réception du signal. C'est pourquoi, des mécanismes de détection d'erreurs sont intégrés à chaque message afin de contrôler facilement leur intégrité. Par ailleurs, avec certaines applications à risques, il devient nécessaire de garantir la réception des messages par le destinataire. Ainsi, le principe d'accusés de réception est le minimum à employer dans ces cas. La sécurisation des données est alors à une couche supérieure dans la diffusion des données. Cependant, ce point constitue de nos jours un défi énorme sur lequel il n'existe pas de réponse générique. Les algorithmes de cryptages qui seront développés devront être léger et avoir un impact très faible sur les échanges de données. Sur un noeud, le gros consommateur d'énergie est le médium de communication [Feeney 01]. Il faut savoir que sur un RCSF, la principale contrainte est l'énergie et le plus gros consommateur est la communication sans fil. C'est pourquoi, différentes recherches développent et proposent des méthodes de communication minimisant les données contenues dans chaque message. Pour cela, les techniques de compression classiques sont employées pour accélérer les échanges et ainsi diminuer les temps d'émission. Enfin, la mobilité des noeuds constitue une importante contrainte à la communication. Les délais de transmission ainsi que la variation de l'environnement entraînent plus de perturbations et plus d'erreurs ce qui demande d'avoir des procédures de validation des messages efficaces mais aussi un meilleur contrôle des transmissions. Un noeud mobile est sujet à un nombre important de paramètres changeant ce qui peut induire une très mauvaise réception. L'exemple le plus parlant est celui des téléphones portables qui consomment beaucoup plus d'énergie lorsqu'on se déplace pour rester associé à une station relais.

1.2.4 Contraintes 'Environnement'

Les contraintes '*Environnement*' rassemblent celles subies par le noeud dans son contexte d'utilisation. Ceci se traduit par des méthodes de fabrication évoluées pour résister aux agressions du milieu surveillé par exemple, ou par une meilleure tolérance aux pannes afin de réduire les interventions humaines. Le noeud doit donc pouvoir fonctionner même en cas de panne en exécutant des opérations simples afin de garantir un fonctionnement minimal le temps pour l'utilisateur d'intervenir sur celui-ci. Traditionnellement, la tolérance aux pannes se traduit par une multiplication des traitements. Dans les applications sensibles comme les avions ou les

navettes spatiales, deux processeurs effectuent la même tâche, pour qu'ensuite leurs résultats soient comparés et en cas de différences, le troisième processeur est activé pour arbitrer (technique de voting processor [Rooks 07]). Ici, les RCSFs ne peuvent se permettre de telle structure afin de minimiser les pannes et autres erreurs. Certaines techniques de tolérance aux pannes dans ces réseaux se traduisent par une coopération entre noeuds afin de garantir une valeur en un point donné. Il est alors possible d'estimer la valeur d'un autre noeud en cas de panne de ce dernier. La tolérance aux pannes peut se traduire aussi par des traitements redondants à l'intérieur même du noeud. S'il n'est pas possible de disposer d'une architecture multi processeurs, il est cependant possible de vérifier la mesure à plusieurs reprises sur le noeud ou avec ses voisins. De ce fait, le principe de calcul collaboratif prend toute son importance et implique d'intégrer cette possibilité au niveau du système d'exploitation et de l'application embarquée. Cette contrainte est très présente sur les réseaux utilisant des noeuds évolués, néanmoins pour des applications utilisant des noeuds de type poussière, elle devient inutile car il y a suffisamment de noeuds déployés pour obtenir l'information redondante. La perte d'un noeud est considérée comme acceptable car il y en a assez pour pallier cette perte. Par ailleurs, dans ce groupe de contraintes, l'impact environnemental du noeud doit être pris en compte. Dans le cas d'utilisation de piles ou de matériaux dangereux pour l'environnement, une démarche de recyclage et/ou de récupération des noeuds doit être prévue. Les noeuds évolués pouvant être facilement récupérés, il n'en est pas de même pour ceux de type poussière. Les matériaux les constituant devront pouvoir être absorbés par le milieu observé à moins de prévoir une méthode de ramassage à l'aide d'un aimant balayant la zone observée par exemple.

1.2.5 Contraintes 'Capteurs'

Enfin, la dernière catégorie, '*Capteurs*', centralise toutes les contraintes liées à l'utilisation d'un capteur. Les coefficients de calibration et autres paramètres sont déterminés et transmis par l'utilisateur au noeud. Bien souvent, ceci se traduit par l'envoi d'une fonction de transfert afin d'exploiter les données issues du capteur. La phase de calibration est donc laissée à l'utilisateur avec les erreurs que cela peut inclure. Toutefois, pour des capteurs simples (course d'un chariot, roue codeuse), il est possible de les calibrer automatiquement. Par ailleurs, pour certains cas d'utilisations des mesures, le noeud et l'utilisateur doivent garantir un niveau de qualité élevée. Par exemple, le suivi de la radioactivité d'une zone contaminée demande une qualité irréprochable des mesures afin de permettre à des personnes de se rendre sur place pour faire des études en toute sécurité. Ainsi, l'application embarquée devra employer des techniques de détection et de fiabilisation des données issues du capteur afin d'obtenir la précision demandée [Tourancheau 09]. Par ailleurs, l'immense variété de capteurs existants, l'interface système capteur doit être le plus générique possible afin de limiter le développement de nouveaux blocs logiciels. Si l'on cherche cette généricité sur les ordinateurs personnels, on trouve alors l'exemple du port série disponible sur toute machine. Grâce à seulement deux lignes de données, une pour la réception, l'autre pour la transmission, le port série dans sa version la plus simple permet de communiquer avec tout périphérique très simplement. Cependant, des interfaces connues comme Inter Integrated Circuit Bus (I^2C), Serial Peripheral Interface (SPI),

Controller Area Network (CAN) ou Analog to Digital Controller (ADC) sont à prévoir pour atteindre une interopérabilité maximale avec les capteurs du marché.

1.3 Conclusion

Pour conclure ce chapitre d'introduction aux RCSFs, il est important de noter l'immense variété d'applications actuelles ou futures où chacune génère des contraintes plus ou moins spécifiques. Une réponse à chacune d'elles demande d'étudier en profondeur les différentes parties d'un capteur, d'un noeud, et d'un RCSF afin de proposer des techniques innovantes et optimales. Pour formaliser un capteur en reprenant le premier schéma de la figure 1.3, la représentation 1.6 détaille les différentes parties essentielles des OIC avec les interactions possibles.

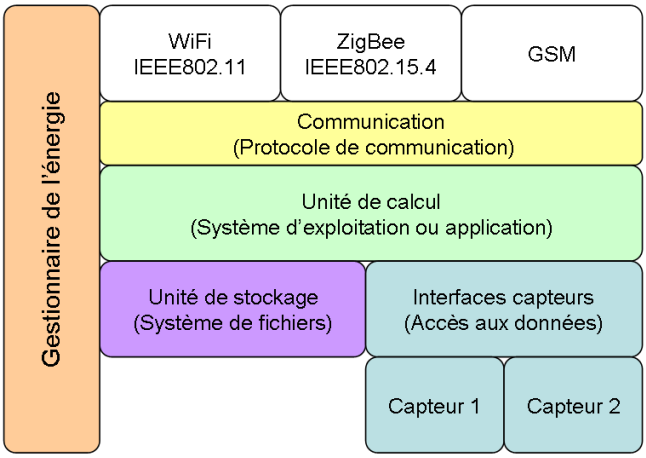

Figure 1.6 – Modélisation d'un noeud capteur

Le coeur du noeud reste bien entendu l'unité de calcul qui déploiera un système d'exploitation pour traiter, envoyer et stocker les données via l'unité de stockage associée. Les capteurs seront utilisés par une interface générique acceptant une liaison de type série par exemple. Cette interface ne se limitera pas seulement à ce type de liaison, d'autres viendront la compléter (I2C, SPI, CAN, ADC). Les données brutes ou pré-traitées seront manipulées par l'unité de calcul à travers les applications embarquées et/ou stocker par l'unité de stockage. Chaque application devra être conçue pour minimiser son empreinte mémoire et son impact sur les ressources matérielles. Ensuite, les échanges de données seront effectués en passant par une couche de communication qui sera en charge de mettre en oeuvre les différents mécanismes pour transmettre et garantir la donnée vers le noeud destinataire. Cette couche sera l'interface entre les applications et le médium de communication sans fil embarquée. Elle sera en charge de manager les périodes de veille du médium ainsi que de gérer et de mettre à disposition les données reçues. Tout ce système sera sous la direction d'un gestionnaire d'énergie dédié ou logiciel qui pourra agir sur tous les paramètres influençant la consommation énergétique du noeud comme la fréquence de fonctionnement et les périodes de veille du médium. On constate alors qu'en dehors des pannes logicielles, l'ensemble des contraintes est lié à la source d'alimentation embarquée ou au médium de communication sans fil. En attendant des solutions technologiques

à ces deux points, le déploiement des RCSFs ne peut être envisagé sans les organiser et sans moyen d'administration. C'est pourquoi, chaque partie du noeud a un besoin spécifique où une réponse efficace et optimale est attendue. Pour résumer et en se basant sur la représentation de la figure 1.6, les noeuds ont des besoins spécifiques concernant :

- *L'énergie* : Un gestionnaire des ressources doit être embarqué pour optimiser l'autonomie du noeud ; il pourra être intégré au système d'exploitation ou être externe via un module externe.
- *La communication* : Il est important d'échanger des informations pour maintenir la cohérence du réseau ; le contrôle de l'intégrité ainsi que le contrôle des échanges sont au coeur de ce besoin.
- *L'organisation* : Un réseau organisé est un réseau qui sera économe énergétiquement, c'est-à-dire qu'il y aura moins de noeuds participant à la diffusion des messages ; la transmission des messages est alors plus rapide grâce à la réduction des sauts des messages.
- *L'interaction avec l'utilisateur* : Le noeud n'est pas seul à prendre des décisions, certains paramètres demandent une intervention humaine ; l'utilisateur doit pouvoir facilement contrôler son réseau à distance.

L'objet de ce mémoire est de présenter une solution permettant à un utilisateur de gérer les noeuds d'un RCSF distant et de proposer une solution simple de visualisation. Cependant, les autres besoins ne peuvent être laissés pour compte et demandent à être intégré dans les travaux. Le contrôle de l'énergie est un élément qui ne peut être tenu à l'écart en raison de son impact important sur l'administration et la durée de vie d'un noeud. De même, les communications et l'organisation du réseau sont des éléments nécessaires pour améliorer la gestion du RCSF. C'est pourquoi, dans le chapitre suivant, nous établirons un état de l'art pour chacun de ces besoins tout en gardant en tête que l'objectif premier est de présenter une solution viable d'administration à distance pour la supervision d'Objets Intelligents Communicants sans fil.

$\sim \star \sim \star \sim \star \sim$

Chapitre 2

Analyse des besoins et état de l'art

Dans le chapitre précédent, nous avons pu voir la grande diversité des applications et des contraintes que subissent les noeuds d'un RCSF. Il est difficile de concevoir une implantation d'un réseau sans avoir des méthodes et outils permettant de prendre en compte les différentes perturbations externes et internes. Les recherches portent donc sur un moyen efficace de comprendre le fonctionnement des noeuds afin de fournir les mécanismes nécessaire au bon fonctionnement du réseau. De nombreuses solutions existent pour répondre aux différents besoins, dont les solutions d'administration à distance.

Ce chapitre présentera un ensemble de méthodes qui ont été conçues pour répondre à ces besoins. On commencera par étudier les solutions permettant à l'utilisateur d'opérer sur les noeuds pour continuer ensuite sur les méthodes complémentaires permettant d'améliorer la diffusion des données dans les réseaux. Le routage et les méthodes d'auto-configuration seront traités dans cette partie comme techniques aidant à l'administration.

2.1 Introduction à l'administration

L'administration d'un RCSF répond à plusieurs objectifs vis à vis de l'utilisateur mais aussi des noeuds. Pour le premier, elle doit faciliter l'accès aux informations des éléments du réseau (réseau/noeud/capteur) pour agir rapidement en cas de problème ou pour modifier des paramètres de fonctionnement. Le second intérêt est de limiter les déplacements humains sur le terrain pour agir sur les différents noeuds. En effet, les RCSFs peuvent être constitués de plusieurs milliers de noeuds ce qui limite fortement les interventions humaines. Enfin, l'administration dans un réseau qu'il soit filaire ou non, intègre des mécanismes permettant l'ordre des noeuds ainsi que leur bon fonctionnement. Cette section présente les caractéristiques d'un protocole d'administration, les différentes couches de réseau pour terminer sur un exemple d'administration dans les réseaux filaires.

2.1.1 Caractéristiques et objectifs d'un protocole d'administration

Les différentes contraintes présentées dans le chapitre précédent, doivent être nécessairement prises en compte dans la définition d'un protocole d'administration. La prise en compte de celles-ci se fera à plusieurs niveaux par différents mécanismes logiciels ou matériels, pour répondre aux critères suivants [Lee 06] :

- *Traitements légers* : Un système d'administration devra être capable de s'exécuter sur les noeuds sans interférer avec le fonctionnement général et sans consommer trop d'énergie supplémentaire. Un ensemble d'opérations plus léger permettra d'augmenter la vie d'un noeud ;

- *Robustesse et tolérance aux pannes* : Les RCSFs sont sujets aux différentes perturbations liées à la communication, à l'énergie limitée et au changement des conditions d'évolution. L'administration doit donc prendre en compte ces caractéristiques pour garantir un fonctionnement optimal quelque soit les conditions en modifiant les paramètres des noeuds si besoin ;

- *Réactivité et adaptabilité* : Le protocole d'administration devrait être capable de récupérer et de s'adapter à l'état courant du noeud, ou de changer sa configuration en agissant sur le rôle du noeud dans le réseau, le niveau d'énergie de fonctionnement, et/ou les performances de communication du médium (portée, puissance d'émission) ;

- *Faible stockage de données* : Le modèle de données utilisé pour représenter les informations de gestion doit être extensible et capable de les accueillir pour effectuer les fonctions d'administration, tout en respectant les contraintes de mémoire des noeuds ;

- *Passage à l'échelle* : La solution d'administration doit pouvoir s'adapter à n'importe quelle taille de réseau.

- *Intrusion* : L'administration doit minimiser son impact sur les performances du noeud et interférer au minimum avec l'application embarquée.

Par ailleurs, la gestion des RCSFs peut être classée en fonction de leur architecture, centralisée, distribuée, ou hiérarchique. Chacune d'elles apporte des avantages et des inconvénients que ce soit sur la réactivité, la communication, ou sur l'accessibilité. La Figure 2.1 représente ce classement. La station de base accède alors à un ou plusieurs noeuds du réseau pour transmettre ces requêtes d'administration.

Les solutions centralisées se basent sur une station de base qui joue le rôle de manager en collectant les informations de chaque noeud tout en contrôlant le fonctionnement général du réseau. Ce manager dispose généralement de ressources illimitées (comparé aux noeuds) et exécute des tâches complexes d'administration, ceci afin de réduire au maximum les traitements embarqués sur chaque noeud du réseau. De plus, il a une vision globale du réseau avec les

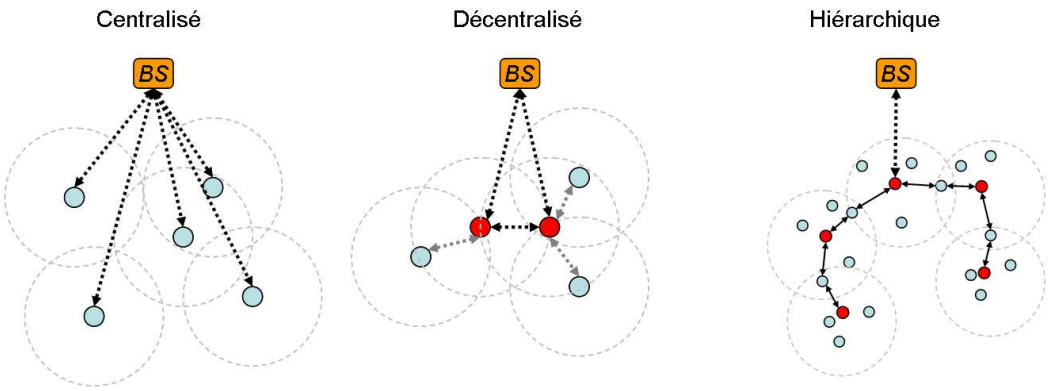

Figure 2.1 – Classification des solutions d'administration

contraintes associées ce qui lui permet de prendre des décisions adéquates. Cette architecture génère donc un goulot d'étranglement avec la station de base, et introduit un trafic important sur les réseaux pour maintenir les informations des noeuds à jour. Dans cette catégorie on retrouve les méthodes Bridge Of the SensorS (BOSS) [Song 05] ou Sensor Network Management System (SNMS) [Tolle 05], qui agissent comme des managers généraux centralisant les données de chaque noeud pour prendre des décisions d'administration.

L'administration des réseaux peut être distribuée sur les noeuds afin de coller au mieux aux contraintes subies. Cette catégorie emploie plusieurs managers pour contrôler une zone du réseau. Chacun d'eux peut coopérer avec un autre manager pour exécuter certaines fonctions générales d'administration. En comparaison aux méthodes centralisées, celles-ci permettent d'améliorer la consommation énergétique des noeuds tout en réduisant les volumes de données échangées. Cependant, ce type d'architecture est complexe et difficile à mettre en place car les différentes fonctionnalités doivent prendre en compte la contrainte de ressources disponibles sur chaque noeud. De plus, la quantité d'informations à stocker par chaque noeud est importante ce qui réduit l'espace mémoire disponible pour l'application embarquée. Dans cette catégorie, on retrouve des protocoles comme Node-energy level management [Boulis 03] ou AppSleep [Ramanathan 05], ou les protocoles basés sur des agents mobiles tels qu'Agilla [Chien-Liang 05] ou Sectoral Sweeper [Erdogan 03].

La dernière catégorie regroupe les protocoles basés sur une architecture hiérarchique comme TopDisc [Deb 01] ou SenOS [Hong 03]. Ici, l'architecture est hybride reprenant certaines caractéristiques des deux catégories précédentes. Les managers intermédiaires sont utilisés pour distribuer les fonctions d'administration mais ne peuvent communiquer directement entre eux. Ils ont en charge un sous réseau et agissent sur celui-ci comme la station de base de l'architecture centralisée. Par ailleurs, une élection est effectuée pour déterminer les managers locaux et pour former les clusters. Cela revient donc à créer une multitude de sous réseaux administrés localement selon une architecture centralisée. Les informations d'administration ne sont envoyées qu'aux managers locaux puis redistribuées localement par celui-ci. En définitive, une telle architecture permet d'administrer des réseaux de grande envergure tout en réduisant les données

envoyées aux noeuds. Cependant, il est indispensable de redistribuer les rôles périodiquement pour éviter d'épuiser les ressources d'un noeud élu gestionnaire 'manager'.

Pour finir, la réactivité d'une solution d'administration peut constituer un classement différent. Une solution passive collecte les informations sur l'état du réseau pour ensuite les traiter a posteriori. Celles basées sur la détection de faute, collectent des données pour détecter et prévenir les erreurs. Ensuite, les méthodes réactives les récoltent pour détecter les événements intéressants et adapter le réseau. Et enfin, les solutions proactives amassent activement les informations d'état pour détecter les événements passés et pour prédire les événements futurs afin d'optimiser les performances du réseau.

2.1.2 Modèle d'interconnexion OSI

En général, dans les réseaux filaires, un message est formaté selon un standard de communication et est traité par différents protocoles avant d'être réellement envoyé au destinataire. Chaque étape d'encapsulation du message peut être décrite par le modèle Open System Interconnection (OSI) composé de 7 couches :

1. La couche "physique" est chargée de la transmission effective des signaux entre les interlocuteurs. Son service est typiquement limité à l'émission et la réception d'un bit ou d'un train de bits continus (notamment pour les supports synchrones).
2. La couche "liaison de données" gère les communications entre 2 machines adjacentes, directement reliées entre elles par un support physique.
3. La couche "réseau" gère les communications de proche en proche, généralement entre machines : routage et adressage des paquets.
4. La couche "transport" manage les communications de bout en bout entre processus (programmes en cours d'exécution).
5. La couche "session" synchronise les échanges et les « transactions », et permet l'ouverture et la fermeture de session de communication.
6. La couche "présentation" est chargée du codage des données applicatives, précisément de la conversion entre données manipulées au niveau applicatif et chaînes d'octets effectivement transmises.
7. La couche "application" est le point d'accès aux services réseaux, elle n'a pas de service propre spécifique et entrant dans la portée de la norme.

Pour les réseaux filaires, ce découpage convient parfaitement et permet de définir un cadre bien précis aux différents protocoles développés pour chacune des couches. Par contre, pour les RCSFs, cette classification doit évoluer pour fournir de meilleures performances énergétiques tout en optimisant les niveaux de développement logiciel. Les couches externes (Application et Physique) sont toujours présentes, par contre, les couches intermédiaires sont regroupées, réorganisées ou supprimées pour mieux correspondre aux besoins des noeuds. Cette réorganisation permet ainsi d'afficher de meilleures performances d'un point de vue énergétique, mais

les débits proposés ne sont pas plus élevés en raison de la QoS qui n'est plus gérée matériellement. L'utilisateur dispose d'un médium de communication "nu" sur lequel il devra adapter son propre protocole de communication.

Concernant l'administration, il est courant de trouver des solutions à la couche applicative comme SNMS [Tolle 05] ou Agilla [Chien-Liang 05]. Cependant, il est possible de descendre dans les différentes couches pour proposer des mécanismes plus ciblés et plus simples. Les protocoles de routage développés pour la couche Réseau peuvent intégrer des fonctionnalités d'administration pour fournir des informations liées aux transmissions et aux perturbations subies par les noeuds. Des mécanismes de synchronisation peuvent être directement intégrés à la couche physique. Cependant, dans ces deux derniers cas, la solution produite devient dépendante de l'utilisation d'un médium de communication ou d'un protocole de routage spécifique. C'est pourquoi, les différentes solutions d'administration sont développées principalement au niveau applicatif afin de permettre une meilleur portabilité quelque soit le médium de communication utilisé. Dans cette optique, nous allons voir le protocole utilisé dans les réseaux filaires pour répondre à la demande d'administration.

2.1.3 Le protocole SNMP : Architecture et fonctionnement

Le protocole Simple Network Management Protocol (SNMP) [Case 90] a été développé à la fin des années 80 pour proposer une solution simple d'administration à distance d'un matériel informatique tel qu'une imprimante, un ordinateur, un routeur et un concentrateur. La première version développée, nommée v1 [Case 90], s'appuie sur des primitives simples (GET, SET) pour accéder aux données et les modifier. La lacune majeure de cette version est le manque de sécurisation des messages échangés ce qui permet à un tiers non autorisé d'effectuer les mêmes opérations qu'un administrateur. Pour pallier à ce manque, une deuxième version (v2c [Case 93]) a introduit une première couche de sécurisation qui restera cependant trop faible. C'est pourquoi, la dernière version (v3 [Blumenthal 02]) introduit un module complet de cryptage des trames et de gestion des utilisateurs. Cependant, la version la plus diffusée reste la version 1 malgré le manque de sécurité, ceci est sans doute dû à la simplicité pour développer un agent version 1 comparée à un agent version 3 qui devra mettre en oeuvre des mécanismes complexes de gestion des utilisateurs. Ensuite, le stockage des données est effectué dans une structure arborescente nommée Management Information Base (MIB) [McCloghrie 91] (Figure 2.2). Défini à l'aide du langage Abstract Syntax Notation 1 (ASN.1) (exemple sur la Figure 2.2), l'utilisateur crée un ensemble de variables basiques (entier, caractère, chaîne de caractères) accessibles, suivant les droits affectés, via les primitives SNMP. On retrouve donc des données sur la description du périphérique, sa localisation ou alors son état courant. Par définition, ce protocole intègre aussi la gestion des communications réseau en récoltant des statistiques ou des paramètres modifiables sur les interfaces réseau et sur les protocoles employés comme TCP, UDP ou ICMP.

Derrière cette nouvelle structure de données, un agent SNMP doit être exécuté pour maintenir à jour l'ensemble des informations dans la MIB. Chaque périphérique supervisé embarque alors son propre agent accessible depuis le réseau. Par ailleurs, SNMP propose différentes mé-

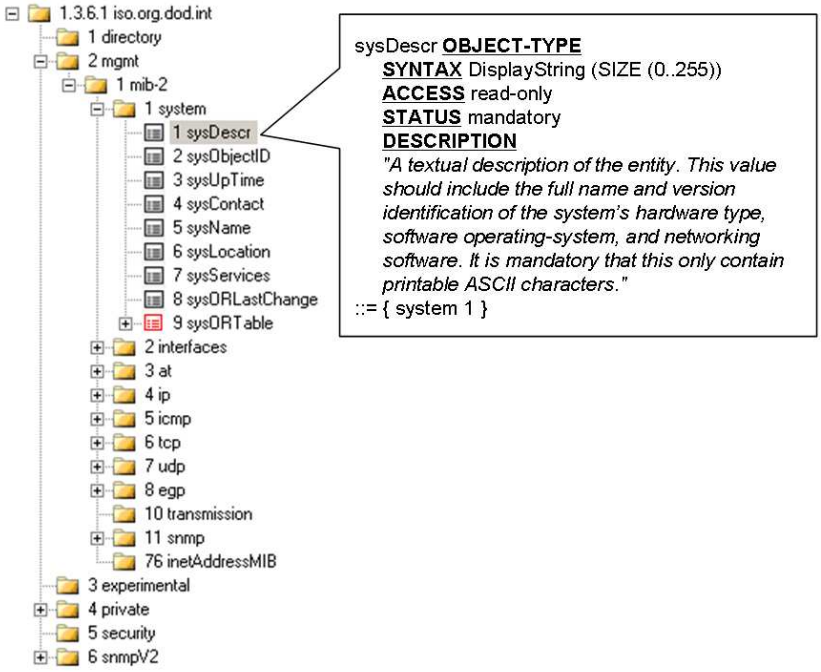

Figure 2.2 – Structure de la MIB-II et exemple de définition d'une chaîne de caractères

thodes pour étendre les fonctionnalités et les données administrées d'un agent. Il est possible de créer une extension à l'agent déjà présent ou un sous agent autonome. La première rend l'extension dépendante de l'agent déjà en place ce qui limite ses possibilités de traitement. Par contre, la seconde technique implémente un sous agent autonome et indépendant de son parent, sur lequel l'intégration de fonctionnalités plus avancées est possible. Par ailleurs, le développement de ce protocole à la couche applicative permet de l'utiliser sur n'importe quel type d'architecture respectant la pile OSI. Avec une popularité très importante et une compatibilité tout système, de nombreux outils de supervision ont été développés ce qui contribue encore aujourd'hui, malgré une mort annoncée depuis des années, à la survie de SNMP dans le monde de l'administration des réseaux informatiques.

2.2 Solutions d'administration existantes

Les RCSFs sont à ce jour l'un des grands enjeux de recherche dans le domaine de l'informatique embarqué. Les problématiques de recherche se posent aussi bien pour transmettre, stocker ou réaliser des mesures sur une plateforme ayant des ressources limitées et sujet à de nombreuses contraintes. Avec de telles caractéristiques, le sujet de l'administration à distance a pris de l'ampleur et a pu voir le développement de quelques méthodes d'administration. La première piste de recherche a été de suivre l'exemple des réseaux filaires en adaptant le protocole SNMP aux contraintes des RCSFs. En parallèle, des solutions d'administration adoptant des structures logicielles spécifiques conçues comme une couche intermédiaire entre la plateforme

matérielle et la partie applicative, ont été développées pour s'adapter au mieux aux conditions difficiles d'évolution des noeuds. L'objet de cette section est de présenter en détails ces deux approches pour apprécier les différentes techniques retenues pour répondre aux contraintes subies par les noeuds.

2.2.1 Méthodes basées sur le protocole d'administration SNMP

Le protocole SNMP, de part sa popularité et sa facilité d'utilisation, semble être un candidat idéal pour superviser un RCSF. C'est pourquoi, on le retrouve dans des protocoles d'administration pour organiser les remontés d'information depuis le noeud vers l'utilisateur via des outils existants. La première piste de recherche a été de reprendre ce protocole pour y intégrer les problématiques liées aux RCSF comme la mobilité des noeuds, la persistance des liens, et la gestion énergétique. On retrouve alors des méthodes comme Ad-hoc Network Management Protocol (ANMP) [Chen 99], Guerrilla management architecture (GUERRILLA) [Shen 02], Spreadsheet based Hierarchical Architecture for MANagement (SHAMAN) [Sethi 01a] ou SNMP Proxy [Yen Yang 08].

Ad-hoc Network Management Protocol

Le protocole ANMP [Chen 99] est un protocole de gestion reprenant le fonctionnement d'un agent SNMP tout en intégrant un sous-arbre de la MIB dédié aux RCSFs pour gérer les contraintes de ressources, la mobilité des noeuds et la stabilité du lien de communication. ANMP implémente le même type de PDU que le protocole SNMP et s'appuie sur le protocole UDP pour transmettre les messages. Les messages perdus ne seront pas retransmis car les données sont envoyées périodiquement et donc en cas de pertes, on attendra la prochaine. Par ailleurs, la gestion de la sécurité des échanges est prise en compte par l'utilisation de la version 3 du protocole SNMP qui intègre un module de cryptage des trames. Afin de disposer d'un protocole utilisant peu de messages, un modèle hiérarchique pour la collecte des données est adapté pour réduire les échanges, car les niveaux intermédiaires de la hiérarchie peuvent rassembler les données (agrégation, compression, moyenne) avant de les transmettre aux couches supérieures de la hiérarchie. Cependant, un problème avec l'utilisation d'une approche hiérarchique dans les réseaux ad hoc est le coût du maintien d'une topologie avec la mobilité des noeuds. Les auteurs ont trouvé un compromis en se basant sur une hiérarchie à 3 niveaux. Au plus bas niveau, on retrouve les noeuds administrés qui sont rassemblés par groupe dans lequel on aura un manager qui dialoguera ensuite directement avec le superviseur du réseau. Il faut rappeler que la hiérarchie utilisée pour l'administration peut être différente de celle qui sera choisie pour le routage des informations par exemple. D'un point de vue administration, le manager et le superviseur sont à un saut, tandis que d'un point de vue routage ils peuvent être séparés de plusieurs sauts. Les groupes seront dynamiques du fait de la mobilité des noeuds. De plus, les managers peuvent évoluer et ne plus être coordinateur d'un groupe.

Pour gérer ce dynamisme, ANMP propose 2 méthodes d'organisation du réseau. La première voit le réseau comme un graphe, où chaque lien entre les noeuds est représenté, et forme les différents groupes de façon à ce que tous les noeuds soient à un ou deux sauts du manager. Dès

qu'un noeud sort d'un cluster, c'est-à-dire quand un noeud ne reçoit plus de message de vie de son noeud maître, il cherche à joindre un nouveau cluster automatiquement. Le noeud avec le plus petit identifiant est choisi pour former un groupe avec ces voisins à un ou 2 sauts. Le second algorithme utilise la position géographique des noeuds pour former les différents groupes en se basant sur une carte spatiale de densité. Le réseau est alors divisé en rectangle de taille fixée par l'utilisateur. Le noeud le plus au centre de chaque zone devient le manager de celle-ci. La mobilité des noeuds modifiera la densité ce qui aura pour effet de relancer le processus de découpage du réseau et donc de faire évoluer les managers.

Enfin, l'ensemble du réseau est constitué de noeuds hétérogènes. Il n'y a pas d'études préalables pour placer des noeuds plus puissants au sein du réseau pour soutenir les fonctions d'administration. Sur chaque noeud, un agent du type SNMP est embarqué pour centraliser les données des noeuds et faciliter leur accès. ANMP utilise les mêmes PDUs et le même protocole de communication (UDP) que le protocole SNMP ce qui le rend entièrement compatible avec les outils existants sur le marché. L'utilisateur peut alors utiliser le même outil pour son réseau filaire et pour son RCSF.

Guerrilla management architecture

Le protocole GUERRILLA [Shen 02] a été conçu de manière à faciliter l'adaptabilité, l'autonomie, et l'hétérogénéité requises dans les réseaux ad hoc. Pour cela, les auteurs proposent d'utiliser un modèle d'administration basé sur le principe client/agence à la place d'un modèle manager/agent. Ce choix autorise des liens point à point entre les différents noeuds d'une agence ce qui améliore la diffusion des données et permet la collaboration entre les noeuds. De plus, ils se basent sur deux couches d'administration pour déployer les différentes fonctionnalités de supervision. La couche haute regroupe un ensemble de managers nomades qui s'adapteront à l'évolution du réseau, posséderont plus d'intelligence, collaboreront ensemble, et serviront de points de liaison avec les noeuds voisins. La couche basse s'appuie sur le mécanisme de sondes actives pour réaliser des traitements localisés dans le réseau.

Le choix des noeuds participants à ces couches se base sur les ressources disponibles. Les noeuds les plus puissants participeront à la couche supérieure en embarquant les fonctions d'administration et fonctionnent alors comme manager pour les autres noeuds. Ils collaboreront ensemble afin de couvrir la totalité du réseau et pour prendre des décisions communes sans aide extérieure. Bien entendu, les noeuds managers vont évoluer au fil du temps en fonction des ressources disponibles, de la topologie du réseau, et de la densité des noeuds. Quand le niveau d'énergie est trop faible, le manager peut décider de migrer vers un noeud plus apte à soutenir les fonctionnalités d'administration. Pour le changement de densité, GUERRILLA prévoit des mécanismes de clonage et de fusion de managers pour correspondre aux besoins du réseau. Par ailleurs, pour décharger les managers des tâches périodiques de collecte de données par exemple, ils ont la possibilité de les déléguer aux noeuds administrés via l'échange de sondes actives. Ces sondes consistent en l'échange de messages contenant des instructions simples à exécuter par le noeud. Ce script navigue alors selon un itinéraire de noeud en noeud avant de revenir au manager qui traitera l'ensemble des données. Les traitements peuvent aller de la simple collecte de données, à l'agrégation de données, ou à la modification de configuration du

noeud. Il interagira directement avec l'agent SNMP local pour effectuer un usage économe de la bande passante limitée.

Les données utiles d'un noeud sont enregistrées dans une structure de données du type MIB, appelée Guerrilla Management Information Base (GMIB), qui ajoute un sous-arbre traditionnel à la MIB accessible par l'agent SNMP local ou par les fonctions de GUERRILLA. La gestion des sondes actives passe par la mise en place d'une interface Execution Environment for Nomadic Manager (EENM) qui contrôle les messages internes entre les différents scripts déployés sur un noeud. Un tel environnement souple permet une mise à jour de certaines fonctionnalités sans modifier le reste de l'application. GUERRILLA est un système modulaire de gestion utilisant une machine virtuelle. Une architecture de gestion avec un environnement spécifique et un langage spécifique peut être un frein pour un simple utilisateur. Toutefois, le principe modulaire autorise l'exécution de différentes applications et traitements sur un même réseau. Les utilisateurs peuvent alors déployer un nouveau module sans avoir à mettre à jour la totalité du logiciel et sans avoir besoin de récupérer les noeuds du RCSF.

Spreadsheet based Hierarchical Architecture for MANagement

Alors que la communauté pour l'administration de réseau, en général, a essayé de concevoir des stratégies de gestion basées sur le concept de gestion par délégation, la communauté SNMP n'était pas encore en mesure d'en profiter parce que les primitives de la délégation n'étaient pas intégrées au cadre de SNMP. C'est grâce à cela, que le protocole nommé SHAMAN [Sethi 01b] a vu le jour. Son objectif est d'incorporer l'administration par délégation aux fonctionnalités de SNMP pour faciliter la gestion de réseaux plus larges. Les objectifs principaux de SHAMAN sont de mettre en place un gestionnaire intermédiaire puissant qui améliore (mais qui est pleinement compatible avec) les fonctionnalités SNMP existantes, fournit des fonctions supplémentaires, supporte la délégation, permet à l'utilisateur de configurer les informations supervisées, et propose un environnement pour le développement rapide d'applications d'administration distribuées.

Les requêtes reçues par un noeud sont traitées par un intermédiaire, Intermediate Manager (IM), avant d'envoyer la réponse à son manager. Chaque IM récolte régulièrement des informations des noeuds proches. La mobilité des noeuds est prise en compte en collectant périodiquement leur position dans le réseau. Ces informations sont traitées ensuite par le manager global pour en déduire et spécifier la zone à laquelle il appartient et ainsi lui assigner un IM. Pour dialoguer entre les noeuds et les IM, le langage Spreadsheet Scripting Language (SSL) est utilisé pour réduire le volume de données échangées. Ce langage proche du langage assembleur est défini de manière à faciliter la définition de la topologie du réseau par les utilisateurs tout en restant facilement compréhensible. En outre, il définit une syntaxe simple et légère pour envoyer des événements, des données ou des commandes. On retrouve alors des fonctions simples d'agrégation de données et des fonctions plus complexe pour la collecte de données comme la commande "poll" qui permet d'activer la mise à jour automatique d'une valeur.

SHAMAN est basé sur des événements de différents types simples ou complexes. Les événements simples regroupent essentiellement ceux représentant les primitives du protocole SNMP et ceux de la plateforme utilisée (*mgrget, mgrset, eventget, timer, activate, deactivate...*). Mgr-

Get et MgrSet correspondent à la réception d'une requête SNMP GET ou SET. Les événements spécifiques sont définis par l'utilisateur à l'aide des événements simples pour remplir une fonction complexe. L'ensemble des événements simples cités précédemment à l'exception d'activate et deactivate, ne peuvent être générés par l'utilisateur. Par contre, dans la catégorie de base, les événements *value change, event occured, invalid value* peuvent être générés directement par les scripts SSL déployés.

Ainsi, l'IM centralise toutes les données vitales des noeuds, et répond aux demandes formulées par un utilisateur. Il convient alors de disposer de noeuds plus robuste et plus performant pour embarquer les traitements supplémentaires causés par les managers intermédiaires. De plus, chaque noeud du réseau doit embarquer de quoi traiter les sondes actives (scripts SSL) au cas où le manager ait besoin de décharger certaines fonctions de collecte de données par exemple.

SNMP Proxy

Dans les précédentes méthodes, l'agent SNMP était utilisé pour collecter les informations d'un noeud. Les différentes requêtes sur les noeuds ont un impact négatif sur les communications et sur l'autonomie des noeuds. La taille des paquets SNMP est importante ce qui consommera plus d'énergie pour les envoyer et augmentera grandement le trafic. Une solution à ces problèmes serait d'utiliser un agent SNMP centralisé qui s'occupera de mettre à disposition les données collectées. Pour cela, la méthode SNMP Proxy [Yen Yang 08] propose d'intégrer une passerelle SNMP pour sauvegarder l'ensemble des données d'un noeud. Les différentes remontées d'informations du RCSF sont alors stockées dans un fichier de log qui sera ensuite retraité pour alimenter l'interface SNMP. La figure 2.3 présente le fonctionnement de cette solution. L'application visée par les auteurs est dans le milieu hospitalier pour suivre les patients à distance. Chaque noeud relève la luminosité près du patient, son rythme cardiaque ainsi que son taux d'oxygène. Les données sont ensuite regroupées vers une passerelle et traitées par une application spécifique CodeBlue [Malan 04]. Le module SNMP vient ensuite récupérer et traiter les fichiers de log de cette application pour compléter les données de sa MIB.

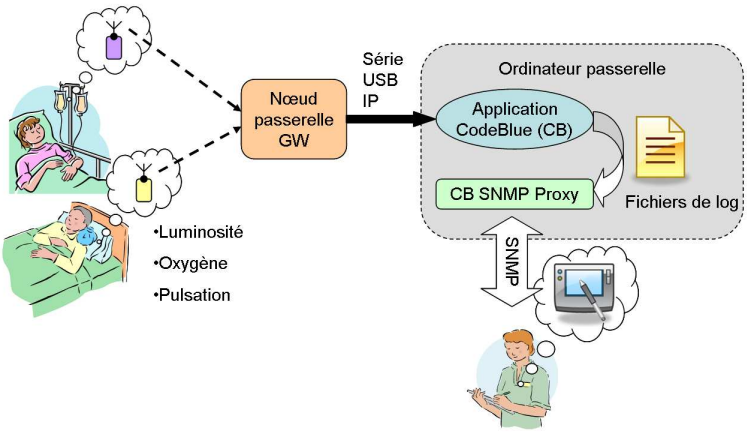

Figure 2.3 – Solution d'administration SNMP Proxy

Les informations extraites des fichiers de log sont insérées dans le prototype à des OIDs connus de la MIB-II. En résumé, cette solution d'administration est directement liée à une application spécifique et ne permet pas à l'utilisateur de modifier des paramètres à distance. L'administration est alors unidirectionnelle [Yen Yang 08]. De plus, l'utilisation de fichier de log altère les temps de mise à disponibilité des informations liées aux patients. Néanmoins, avec un agent SNMP central, le trafic sur le RCSF est alors plus faible du fait de l'utilisation d'un protocole de communication mieux adapté à leurs contraintes notamment sur la taille des messages.

Synthèse

Les méthodes présentées précédemment montrent l'intérêt porté au protocole d'administration SNMP. Malgré une mort annoncée depuis de longues années, le protocole SNMP trouve dans les RCSFs une nouvelle jeunesse. La large diffusion de ce protocole dans le domaine filaire a permis de disposer d'outils de gestions aboutis au fil du temps. Il était donc légitime de voir son utilisation pour les RCSFs. Pour chaque méthode, un agent SNMP est utilisé pour centraliser les données du noeud et les mettre à disposition de l'utilisateur. Chaque noeud possède son propre agent à part pour la dernière solution où il y en a qu'un seul agent pour tout le réseau. L'accès à leurs données peut alors se faire à distance par un ordinateur ou un PDA via une connexion réseau. La Table 2.1 compare les différentes méthodes présentées pour pouvoir en tirer des conclusions en vue d'une éventuelle utilisation de ce protocole dans une nouvelle solution d'administration.

Table 2.1 – Caractéristiques des solutions d'administration basées sur le protocole SNMP

	ANMP	GUERRILLA	SHAMAN	SNMP Proxy
Accès aux données	+ +	+ +	+ +	+
Réactivité	+	+	+	+
Dynamisme	+	+ +	+	- -
Volume de données échangées	- -	- -	- -	+
Empreinte mémoire	- -	- -	- -	+ +
Données administrées	-	+	-	- -

Pour les trois premières méthodes, l'accès aux données est direct ce qui permet de modifier les paramètres directement sur les noeuds. De plus, les données administrées pour ces méthodes sont variées allant de la gestion de la mobilité des noeuds, à la gestion énergétique en passant par l'attribution de rôle dans le réseau. Malheureusement, pour ces méthodes, l'empreinte mémoire est trop élevée pour un déploiement sur des RCSFs et l'utilisation des requêtes SNMP sur le réseau engendre un trafic important usant la source énergétique des noeuds. Contrairement, à celles-ci, la dernière solution présentée optimise les échanges en utilisant un agent SNMP

centralisant les données. L'agent SNMP qui sera déporté sur un serveur plus performant libérera de l'espace mémoire sur les noeuds. La contrepartie de ceci est l'introduction d'un point qui centralisera aussi les problèmes de collision, de tolérance aux pannes et de surcharge réseau.

Pour résumer, le protocole SNMP présente des avantages non négligeables concernant l'accessibilité aux données managées et la diversité des outils existants. Malheureusement, l'usage d'un agent SNMP par noeud n'est pas envisageable. La piste lancée par SNMP Proxy permet un compromis permettant un meilleur accès aux données tout en préservant les noeuds des lourdes requêtes SNMP. Cependant cette dernière ne permet pas un accès bidirectionnel vers les noeuds. L'architecture de la nouvelle solution d'administration de ce type serait basée sur un agent centralisant les informations remontant des noeuds et pouvant les modifier directement sur les noeuds.

2.2.2 Middlewares ou Intergiciels

Nous avons pu voir dans la partie précédente que le protocole SNMP a été très utilisé dans les RCSFs pour leur administration. Son utilisation était principalement motivée par les nombreux outils de supervision disponibles et l'adoption de ce protocole sur tous les périphériques et systèmes reliés en réseau. Néanmoins, une autre catégorie de solutions d'administration existe, les middlewares ou intergiciels. Par définition, un intergiciel a pour fonction principale l'interfaçage entre les dispositifs matériels et les applications d'un noeud. Plus précisément, son rôle est d'offrir des services standardisés pour une manipulation efficace et adaptée des ressources matérielles disponibles (Figure 2.4). L'exemple type est la machine virtuelle JAVA utilisée sur nos ordinateurs pour exécuter des scripts locaux ou distants. L'utilisateur développe une application à travers un script, qui sera ensuite interprété par la machine virtuelle à l'exécution.

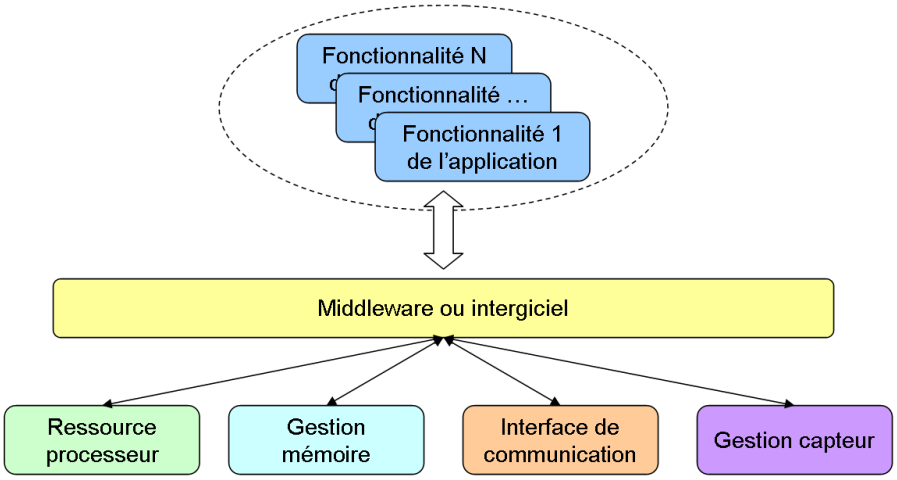

Figure 2.4 – Architecture d'une application basée sur un intergiciel

Les différents types d'intergiciels peuvent être classés en plusieurs catégories [Hadim 06] orientées vers une machine virtuelle, des agents mobiles ou une base de données. Chacune de

ces méthodes impliquent alors de disposer d'une structure spécifique pour traiter les requêtes et pour exécuter de nouvelles fonctionnalités. Les sous parties suivantes présenteront plus précisément ces 3 catégories majeures de middlewares.

Type machines virtuelles

Le premier groupe de solution est le plus flexible et regroupe les machines virtuelles, les interpréteurs et les agents mobiles. Il permet aux développeurs d'écrire des applications sous forme de petits modules indépendants. Les agents mobiles représentent ici une suite d'instructions qui seront exécutées par le noeud. Les agents mobiles et les agents SNMP sont très différents, le premier sert à échanger des commandes tandis que le second représente une application sur une machine qui traite l'ensemble des requêtes reçues. Le système injecte et distribue les agents mobiles via le réseau en utilisant des algorithmes adaptés de façon à ce que l'énergie globale consommée et l'utilisation des ressources soient réduites au minimum. La machine virtuelle interprète ensuite ces différents modules. Cette approche souffre toutefois de la surcharge de travail pour traiter les instructions reçues.

Par exemple, Maté [Levis 02] (Figure 2.5) implémente une machine virtuelle pour exécuter des modules logiciels sur un noeud afin de répondre à la problématique d'administration. Cette solution met l'accent sur la nécessité de nouveaux paradigmes de programmation pour surmonter les contraintes telles que la bande passante limitée et la forte consommation énergétique des activités sur le réseau. Maté propose alors un large spectre de re-programmation, du simple ajustement de paramètres jusqu'à l'envoi de mise à jour de programme complet via la machine virtuelle. Il est défini comme un interpréteur de code fonctionnant sous TinyOS [Levis 05], un système d'exploitation (ou Operating System (OS)) conçu pour être embarqué sur des noeuds ayant de faibles ressources. Développé sur cet OS, Maté devient alors difficile à mettre en oeuvre sur d'autres systèmes ou d'autres plateformes.

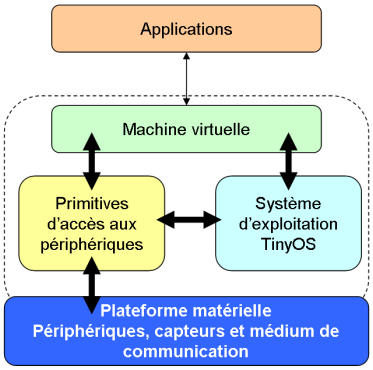

Figure 2.5 – Architecture de l'intergiciel Maté

En deuxième exemple, Magnet [Barr 02] est un intergiciel utilisant une machine virtuelle et faisant parti de MagnetOS [Barr 02], un système d'exploitation adaptatif gérant son énergie et spécialement conçu pour les RCSFs et les réseaux ad-hoc. Il se constitue d'une couche

définie comme une image simple du système (Single System Image) qui fournit une abstraction de haut niveau de l'hétérogénéité du réseau. Cette vision permet de voir le réseau comme une seule et même machine virtuelle JAVA (Figure 2.6). Ce système comprend un composant statique capable de récrire les applications JAVA de l'utilisateur en objets ou modules définis par le système d'exploitation MagnetOS. Ensuite, chaque composant est diffusé sur le réseau accompagné d'instructions spécifiques pour rester dans la sémantique des messages. A partir de là, un élément actif et dynamique sur chaque noeud se charge de surveiller la création, l'utilisation, et la migration des objets fournissant de nouvelles fonctionnalités à l'application. Concernant les performances, MagnetOS offre une flexibilité pour les programmeurs pour régler explicitement le placement et la migration d'objets. Cela leur permet de réduire les communications réseau en plaçant les objets au plus près des sources de données, par exemple. MagnetOS offre un algorithme robuste de gestion de l'énergie qui s'appuie sur la migration d'objets de la même application vers les noeuds qui sont proches. Ce mécanisme réduit alors la consommation énergétique de l'application ce qui augmente l'autonomie des noeuds.

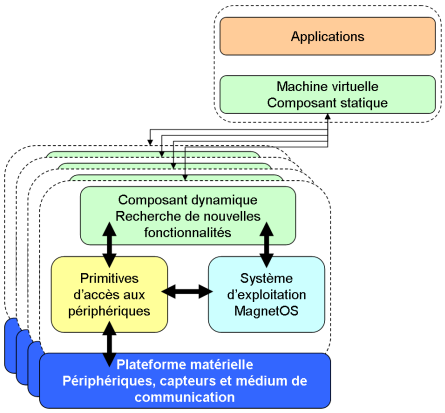

Figure 2.6 – Architecture de l'intergiciel Magnet associé au système d'exploitation MagnetOS

Type programmation modulaire

Le deuxième groupe d'intergiciel s'appuie sur une programmation de type modulaire. La clé de cette approche est que les applications doivent être modulaires pour faciliter leurs injections et leurs distributions dans le réseau en utilisant des agents mobiles, ou codes mobiles. La transmission de petit volume de données permet entre autres de consommer beaucoup moins d'énergie que l'échange d'une application entière. On retrouve alors la technique Impala [Liu 03] qui offre des mécanismes pour la mise à jour de RCSF. L'aspect automatique de cette solution permet d'augmenter la tolérance aux pannes et l'auto-configuration du réseau. Par contre, la nature des instructions ne permet pas de l'utiliser sur des noeuds utilisant des plateformes matérielles hétérogènes. Il implémente une couche logicielle intermédiaire asynchrone basée sur des événements afin de gérer les agents mobiles diffusés sur le réseau. Ces agents sont compilés en binaire avant d'être déployés sur le réseau. Cette approche assure l'adaptation de l'application et peut automatiquement distinguer la configuration nécessaire des paramètres ou les logiciels

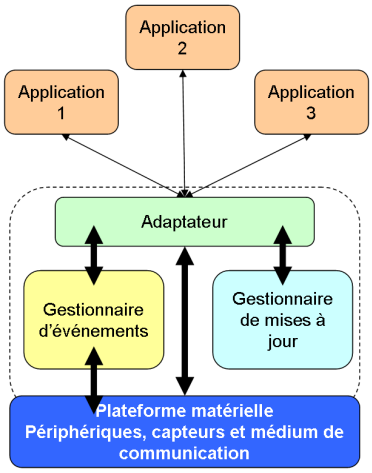

Figure 2.7 – Architecture de l'intergiciel Impala

utilisés. L'utilisateur a alors la possibilité de déployer de nouveaux protocoles à tout moment, et de basculer entre eux à volonté. Impala s'appuie sur deux couches logicielles (Figure 2.7), une pour l'application et les protocoles de communication et une seconde qui regroupe la gestion bas niveau des agents mobiles avec les événements associés, l'adaptateur, et la mise à jour. En résumé, le gestionnaire d'événements traite ce qui concerne les échanges avec la partie matérielle et les différents modules ; l'adaptateur offre une interface de liaison pour les agents mobiles ; le module de mise à jour prend en charge la diffusion des mises à jour dans le réseau en prenant en compte la mobilité des noeuds, la charge de la bande passante, et la gestion mémoire des codes reçus. La gestion des versions des applications se base sur des numéros de version qui sont diffusés avant chaque mise à jour. Ensuite, le noeud fait la demande des modules à mettre à jour ce qui réduit grandement la charge du réseau pour la mise à jour d'une simple fonctionnalité sur tous les noeuds.

Type base de données

La prochaine catégorie propose de voir l'ensemble des noeuds comme une base de données virtuelle sur laquelle il sera possible d'appliquer différentes requêtes du type Sructured Query Language (SQL) par exemple. Elle offre une interface facile à utiliser permettant à l'utilisateur d'envoyer ses requêtes pour extraire les données intéressantes. Toutefois, cette approche ne permet pas d'avoir des applications temps réel car il manque la synchronisation spatio-temporelle entre les requêtes. Cougar [Yao 02] (Figure 2.8) fait partie de cette catégorie. Cet intergiciel introduit une nouvelle dimension dans ce domaine en adoptant une approche orientée base de données. L'ensemble des capteurs est représenté dans une base virtuelle où chaque noeud est une instance. Par ailleurs, Cougar implémente les opérations d'administration de RCSF sous forme de requêtes basées sur le langage SQL. Ce système a demandé la mise en place d'une base de données et des requêtes associées. Les données sauvegardées sont représentées par des relations qui incluent l'ensemble des noeuds participants dans la base de données et les caractéristiques

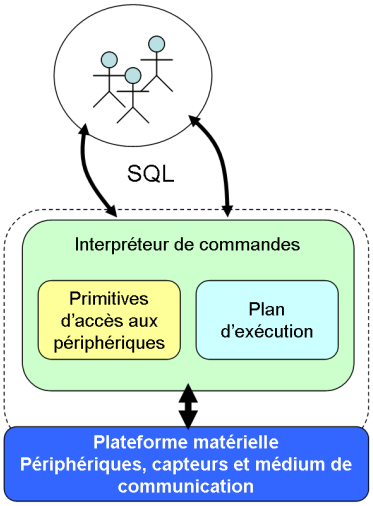

Figure 2.8 – Architecture de l'intergiciel Cougar

des noeuds et de leur environnement physique.

En deuxième exemple, le module TinyDB [Madden 05], inclus dans le système d'exploitation TinyOS, est un système de requêtes pour extraire les informations d'un réseau de capteurs utilisant cet OS. En général, les applications utilisant TinyOS obligent l'utilisateur à écrire en langage C les fonctions pour extraire les données. Cependant, TinyDB soulage l'utilisateur de cette complexité en proposant une interface facile à utiliser basée sur le langage SQL pour extraire les données intéressantes des noeuds. Les requêtes utilisent des opérations simples pour spécifier le type de données aussi bien que le sous ensemble de noeuds intéressants. Pour cela, une table virtuelle est maintenue où les colonnes contiennent les informations des capteurs comme le type, l'identifiant, ou le niveau de batterie. L'ajout d'un capteur se résume alors à l'ajout d'une ligne dans cette table. D'un point de vue communication, TinyDB utilise une approche par inondation contrôlée pour diffuser les requêtes dans le réseau, et maintient un arbre de routage pour accéder à l'utilisateur final.

Synthèse

Les solutions d'administration basées sur des middlewares offrent une forte modularité et une grande adaptabilité. L'utilisateur a la possibilité d'introduire de nouvelles fonctionnalités à volonté dans le cas des machines virtuelles et de la programmation modulaire. Dans ces deux cas, il est nécessaire d'embarquer une interface générique qui aura pour objectif de traiter les différents modules présents sur le noeud et d'organiser les échanges entre eux. Les traitements seront donc nécessairement plus long le temps de traiter le script avant son exécution effective. Ce point constitue un inconvénient important à l'utilisation de ces architectures pour les RCSFs où les noeuds ont un espace mémoire faible. Une grande partie sera consommée par les différents traitements internes au détriment de l'application de l'utilisateur. La dernière catégorie s'appuyant sur l'architecture base de données permet de fournir un système performant

Table 2.2 – Caractéristiques des middlewares

	Machines Virtuelles	Programmation modulaire	Base de données
Modularité	+ +	+ +	-
Réactivité	+	+	+
Charge de calcul	- -	- -	+
Empreinte mémoire	- -	-	+
Administration	+	+	- -

de requêtes et de stockage des données. Malheureusement, ces intergiciels ne permettent pas d'avoir un fonctionnement temps réel et n'intègrent pas les problématiques d'administration. Par contre, les middlewares des différentes catégories proposent des méthodes intéressantes à prendre en compte pour le développement d'une solution d'administration. La synchronisation et la diffusion des agents mobiles peuvent faire partie d'une solution de mises à jour centralisée par exemple. Une comparaison des différentes catégories est présentée par la la Table 2.2. Les critères de comparaison sont le degré de modularité, l'empreinte mémoire, la charge de calcul, et le degré d'intégration de l'administration dans leur structure. Même si l'on présente ces 3 catégories pour l'administration, certaines ne proposent qu'une faible capacité d'administration.

La Table 2.2 met en évidence une faible intégration de l'administration dans les intergiciels présentés. Même si dans IMPALA, on trouve un module de gestion de mise à jour des codes, les fonctionnalités d'administration sont le plus souvent représentées par un agent ou un script exécuté par l'intergiciel comme s'il s'agissait d'une application quelconque. Il est donc peu intéressant de prendre ce type d'architecture pour l'administration d'un RCSF à moins d'intégrer à la base les fonctionnalités de gestion. Par ailleurs, les intergiciels sont basés sur un OS spécifique bien souvent lié à une plateforme matérielle. Il est donc difficile d'envisager le portage de ces techniques sur un OS différent ou sur une architecture matérielle différente.

2.2.3 Préservation de la ressource énergétique

Réduire la quantité d'informations échangées constitue une première phase intéressante pour limiter la consommation énergétique. Toutefois, l'autonomie d'un noeud ne se résume pas uniquement à la quantité de données échangées. En effet, la plateforme matérielle use aussi la source énergétique. Pour en réduire l'effet, le premier réflexe est donc de commander l'extinction des parties du noeud qui ne servent pas ou très peu. Il est alors possible d'avoir un noeud complètement éteint pendant de longues périodes d'inactivité.

AppSleep [Ramanathan 05] implémente une fonction d'administration de l'énergie au niveau de la couche Application de la pile OSI pour permettre d'organiser les périodes de sommeil en se basant sur un flux de paquets au lieu d'échanger des messages individuels. Typiquement, il garde éveillés les noeuds qui traitent un transfert de données fragmentées et éteint les autres.

AppSleep est résistant aux variations de la densité de ces voisins et il s'appuie sur un schéma qui permet aux applications de se configurer pour répondre aux divers besoins de latence, tout en maximisant l'efficacité énergétique. Une limitation d'AppSleep est que, pour tirer profit de sa méthode d'économie d'énergie, les applications doivent être capables de tolérer des délais de communication plus longs. Puisque les opérations d'AppSleep dépendent de la couche de routage, le protocole de routage choisi devra tenir compte de ces caractéristiques : lorsqu'une route active change, le protocole de routage doit être capable d'établir de nouvelles liaisons au début de la période de réveil du noeud ; il doit aussi veiller à ce que les trajets choisis pour les transferts de données restent actifs durant toute la période d'activité du noeud.

SenOS [Hong 03] est une machine d'états finis basée sur un système d'exploitation embarquant un système de contrôle de l'énergie. Elle prend comme hypothèse qu'il y a des noeuds redondants dans chaque cluster pour alterner les opérations afin de prolonger la durée de vie des clusters. SenOS garde seulement un noeud éveillé dans un cluster pendant un temps, tandis que les autres sont tenus au sommeil. Pour permettre cela, cette méthode utilise le Dynamic Power Management (DPM), une technique d'administration de l'énergie basée sur un algorithme adaptatif, pour contrôler l'extinction des noeuds si nécessaire. DPM offre des mécanismes pour déterminer les changements d'états en se basant sur les événements observés afin de maximiser l'efficacité énergétique du noeud. SenOS exprime les transitions issues de DPM en un modèle de machine d'états finis et exécute la gestion de l'énergie sur les noeuds du réseau en se basant sur celui-ci. De plus, il peut être étendu à d'autres protocoles d'administration de noeuds, mais la contrepartie est d'avoir le système d'exploitation SenOS sur chacun des noeuds du réseau.

Enfin, il peut être intéressant d'avoir une méthode de contrôle des applications exécutées sur un noeud. Son rôle serait d'accepter ou refuser une exécution en fonction du coût énergétique induit, de sa pertinence, et si elle répond bien aux objectifs de l'utilisateur [Boulis 03]. Dans une approche système, chaque application doit connaître les caractéristiques des autres ce qui implique un surplus de trafic dans le réseau. Pour répondre à cette idée, le mécanisme nommé Node-Energy Level Management [Boulis 03], a été développé pour résoudre le problème de gestion de l'énergie au niveau du noeud à défaut du niveau système, là où il est encore possible de prendre des décisions sur les tâches à accepter. Les applications n'ont alors plus besoin de se connaître directement. La solution proposée évalue trois paramètres, énergétique, gain et politique d'accès, avant d'accepter ou refuser une tâche. Les attributs énergétiques sont une liste de services, chacun contenant les paramètres qui définissent l'utilisation du service et le coût énergétique d'une tâche spécifique. Ces valeurs sont prédéfinies au moment de la création de la tâche. Les paramètres liés au gain d'une application spécifient la priorité définie par l'utilisateur et les nouvelles fonctionnalités de l'application. Par exemple, à la réception d'une application, le noeud évalue son coût énergétique ainsi que son gain. Ensuite, le résultat obtenu est comparé à la politique d'acceptation définie dans le réseau ainsi qu'aux ressources disponibles sur le noeud. Pour finir, le mécanisme Node-Energy Level Management décide de l'intérêt de cette tâche vis à vis des objectifs recherchés par l'utilisateur.

2.2.4 Synthèse générale

Comme nous avons pu le voir, il existe différentes approches pour administrer les RCSFs. Qu'elles soient basées sur un protocole issu du domaine filaire comme SNMP, un protocole propriétaire ou sur des intergiciels, une contrainte demeure : la gestion des ressources énergétiques et matérielles. Les méthodes basées sur le protocole SNMP par exemple, demandent de disposer de noeuds plus performants afin d'accueillir une partie ou tout un agent. Ensuite, des méthodes comme TinyDB, Magnet ou Maté, utilisant un système d'exploitation spécifique (TinyOS, MagnetOS), ont une portabilité et interopérabilité très faible. Même si le langage utilisé est proche du langage C, un important travail de réécriture doit être envisagé avant de pouvoir les utiliser sur un autre OS.

Pour faire un bilan de ces différentes techniques et approches de l'administration, le tableau 2.3 reprend leurs avantages et inconvénients. Pour des solutions basées sur un existant, la majeure partie du travail est de le rendre compatible avec les contraintes issues des RCSFs. Or, l'adaptation d'un protocole existant et développé dans un contexte spécifique est un exercice périlleux, plus ou moins stable et surtout peu optimisé à la plateforme utilisée. Dans les approches utilisant SNMP, un agent a été adapté pour être embarqué sur les noeuds du réseau ce qui oblige à les sur dimensionner et donc à augmenter leur consommation énergétique. De plus, la faible variété de données administrées de cette méthode la rend peu adapté aux cas des RCSFs agri-environnementaux. Cependant, le principe même d'utiliser ce protocole semble très intéressant du fait de son fort encrage dans le monde des réseaux filaires. Les protocoles spécifiques sont, du fait qu'ils soient liés à un système d'exploitation la majeure partie du temps, difficiles à mettre en oeuvre sous d'autres environnements ou systèmes d'exploitation. Les idées qu'amènent chacun d'eux sont innovantes que ce soit sur la gestion de l'énergie, des codes embarqués ou des communications. Les solutions d'intergiciels offrent une bonne modularité du système et des possibilités d'évolution très importantes. Leur inconvénient est bien sûr lié à l'ajout de traitements supplémentaires pour mettre en place leur architecture. Les machines virtuelles ou autres interfaces de liaison ont un coût à la fois mémoire mais aussi processeur pour le traitement et l'exécution des nouveaux traitements reçus.

En définitive, les solutions d'administration présentées dans ce chapitre ont été conçues pour différentes perspectives, surveillance d'un réseau de santé, tolérance aux fautes, gestion du trafic réseau ou des ressources du noeud. Ces systèmes se caractérisent par les efforts portés sur la consommation énergétique, la gestion des ressources mémoire ou processeur, la bande passante utilisée, la tolérance aux fautes ou leur adaptabilité. Néanmoins, aucune des solutions présentées n'offre une vue intégrant toutes les caractéristiques d'un protocole d'administration adapté à ces applications. De plus, certaines d'entre elles lient les fonctions d'administration à celles de l'application. Enfin, les solutions proposées pour l'administration ne répondent qu'à des contraintes bien spécifiques dans un environnement spécifique. Pour les migrer vers d'autres systèmes d'exploitation, un travail important de réécriture est nécessaire. De plus, chaque plateforme est différente ce qui implique autant d'optimisations qu'il y aura de type d'architecture. Le développement d'une couche d'administration générique est un défi très important laissant de nombreuses pistes de recherche inexplorées pour les RCSFs. Enfin, pour l'ensemble de ces approches et techniques, les fonctionnalités d'administration se basent sur le principe de don-

Table 2.3 – Comparatif des différentes approches et techniques d'administration basées sur le protocole SNMP ou sur des intergiciels

	Avantages	Inconvénients
Protocole SNMP (réseau filaire)	– Facile d'utilisation – Compatible tout système	– Sécurité (v1, v2c) – Empreinte mémoire élevée
Méthodes basées sur le protocole SNMP	– Interface utilisateur – Récolte des données	– Peu optimisé – Empreinte mémoire élevée
Intergiciels	– Modularité et évolutivité – Contraintes bien prise en compte	– Ajout d'une charge de calcul – Empreinte mémoire moyenne

nées à la demande, c'est-à-dire que pour connaître un paramètre, le superviseur devra produire une requête par paramètre souhaité. L'énergie dépensée devient alors très importante et rend ces méthodes intrusives. L'impact sur la vie du noeud et sur le fonctionnement de l'application embarquée est alors important. Il serait donc intéressant de minimiser ces différentes requêtes avec des méthodes simples comme la compression, l'estimation ou le diagnostic indirect. Dans ce dernier cas, la connaissance du contexte d'évolution ainsi que des plages de valeurs attendues permettrait, entre autres, de déduire des informations sans nécessairement les demander.

2.3 Méthodes participant à l'administration

Les solutions d'administration présentées posent toutes la même hypothèse : le réseau dispose de tous les outils nécessaires pour diffuser les informations d'un point A vers un point B. De plus, l'auto-configuration des réseaux d'un point de vue routage est perçue comme acquise par toutes ces méthodes. Néanmoins, nous avons pu voir que l'administration ne peut pas être mis en place sans des outils de routage de l'information ou d'auto-configuration des réseaux. L'objectif de cette section est de présenter ce qui se cache en dessous de l'administration sans quoi, il ne serait pas possible d'échanger des messages dans les réseaux.

2.3.1 Organisation d'un réseau et de ses communications

Les ressources étant limitées, il est important d'organiser au mieux les capteurs afin d'optimiser les temps de transfert et de fonctionnement. Pour cela, il existe différentes méthodes pour permettre à un RCSF de s'organiser ou de se réorganiser. Ainsi, il est courant de retrouver les techniques d'organisation utilisées pour les réseaux filaires. On distingue alors différentes topologies de réseau :

- *Étoile* : Tous les noeuds sont reliés en un même point (concentrateur). La défaillance d'un noeud n'interrompt pas l'utilisation du réseau mais une anomalie sur l'équipement central entraîne sa perte totale.
- *Anneau* : Tous les noeuds sont reliés "en série". La perte d'un noeud rompt le réseau et donc le transfert de données. Lors d'une collision, les deux messages sont perdus.
- *Maillage ou point à point 'Peer to Peer'* : Tous les noeuds sont reliés entre eux. Ceci permet de fournir différents itinéraires possibles pour diffuser une information d'un point A à un point B. Par ailleurs, une telle architecture permettra de diffuser facilement des informations en augmentant sa disponibilité sur chaque noeud.
- *Arbre* : Les différents noeuds sont reliés sous forme d'arbre. Chaque noeud n'est voisin que d'un seul autre noeud qui est son parent.
- *Cluster* : Le réseau est divisé en plusieurs zones administrées localement par un noeud défini comme maître afin de réduire les informations échangées dans le réseau. Cette méthode s'appuie à la fois sur une topologie étoile pour les clusters, et sur une topologie arborescente ou maillée pour la liaison entre les différents noeuds maîtres.

A travers ces différentes topologies, on constate alors que l'utilisation de clusters dans un RCSF constituera une solution efficace pour diffuser de l'information tout en réduisant les noeuds y participant. Le découpage en anneau ne peut être envisagé à cause des contraintes de communication liées aux transmissions sans fil mais aussi à cause des risques non négligeables de pertes d'un noeud. Enfin, le maillage de réseau a l'avantage de proposer différents chemins dans le réseau mais oblige chaque périphérique à être en permanence à l'écoute donc actif. Pris séparément, les découpages en clusters, maillé ou arborescent ne présentent que peu d'avantages. Par contre, en couplant le premier avec un des deux suivants, de nombreux avantages apparaissent comme la possibilité de n'utiliser qu'une partie des noeuds pour échanger des données pendant que les autres sont endormis, ou d'étendre le réseau sur de vastes distances. Tous les noeuds ne sont plus nécessairement à portée de la station de base principale ce qui permet une couverture du réseau plus importante. A partir de ces différentes topologies, il existe des méthodes d'auto-configuration telles que LEACH [Heinzelman 00] ou TopDisc [Deb 01], qui utilisent le principe de cluster avec une base en étoile ou arborescente. Chaque cluster est formé d'un noeud dit maître ou manager local qui embarquera des fonctionnalités avancées pour maintenir et organiser les communications avec les noeuds qu'il manage appelés noeuds esclaves. Ces derniers ne peuvent communiquer qu'avec leur noeud maître et c'est à lui de faire suivre les messages à l'extérieur du cluster si besoin. Pour des applications, certains d'entre eux sont élus pour relayer les informations entre les différents clusters jusqu'à la station de

base. Leur nombre varie en fonction de la redondance des liens désirés entre les clusters afin d'avoir une meilleure transmission des données en évitant de n'user qu'un seul relais. Les messages transiteront alternativement par chaque relais, et en cas d'envois simultanés, ils pourront utiliser deux chemins différents pour éviter les collisions.

Principales méthodes d'auto-configuration

La première phase du déploiement d'un RCSF est une phase d'organisation afin de déterminer une structure de communication. Dans les réseaux filaires, cette phase est masquée par l'administrateur de réseau qui détermine en amont les liens de communication en reliant chaque machine à un concentrateur ('switch' commutateur, routeur ou autre). L'architecture généralement retenue dans ce cas est une architecture hiérarchique en étoile, c'est-à-dire que chaque concentrateur est connecté à un autre et chaque machine est reliée à un concentrateur. Pour les RCSFs, les contraintes de communication ainsi que leur taille ne permettent pas de définir à l'avance une topologie type. Il est donc nécessaire de disposer d'un moyen d'auto-configuration pour déterminer et faire évoluer l'organisation des noeuds dans le réseau. De plus, un réseau organisé sera un réseau qui optimisera les chemins de communications et qui pourra faire quelques économies d'énergie en n'utilisant que les noeuds strictement nécessaires à la diffusion d'un message.

L'un des plus connus est nommé Low Energy Adaptative Clustering Hierarchy (LEACH) [Heinzelman 00]. Il se base sur la formation de cluster relié en étoile à un saut de la station de base. Il se définit comme un protocole autonome et adaptatif de clusterisation qui utilise un tirage aléatoire des noeuds maîtres afin d'uniformiser leur énergie. Comme tout cluster, LEACH définit un maître qui va servir de station relais entre le superviseur et ses voisins. Un premier réflexe serait de le fixer au démarrage mais il est facile de constater que ce mode épuisera très rapidement ses ressources énergétiques. C'est pourquoi LEACH intègre un algorithme qui permet de redéfinir périodiquement les noeuds maîtres. Ce changement s'effectue sur la base des ressources disponibles sur les noeuds, c'est-à-dire, que la station de base envoie les rôles de chaque périphérique périodiquement en fonction de l'état des ressources disponibles à cet instant. Par la suite, les autres noeuds se rattachent au noeud maître qui demande le minimum d'énergie pour communiquer, typiquement celui qui sera le plus proche. La phase suivante consiste à réguler l'utilisation du médium de communication en les éteignant. Périodiquement, chaque noeud l'allumera pour envoyer ces informations puis l'éteindra. Le manager local définit la périodicité des remontés d'informations en fonction de la fréquence d'envoi désirée vers le superviseur. Toutes les données issues des capteurs sont traitées ou agrégées afin de réduire le volume de données échangées entre les maîtres et le superviseur. La Figure 2.9 présente un exemple de topologie issue du protocole LEACH sur un réseau distribué aléatoirement. En définitive, l'avantage de LEACH est de permettre une reconfiguration automatique du réseau sous forme de cluster en prenant en compte les coûts de communication entre chaque noeud. La contrepartie est que le superviseur doit avoir accès à tous les noeuds. Par ailleurs, LEACH suppose que les liens de communication sont symétriques et que chaque noeud dispose d'un moyen efficace d'évaluer son énergie et sa puissance de réception.

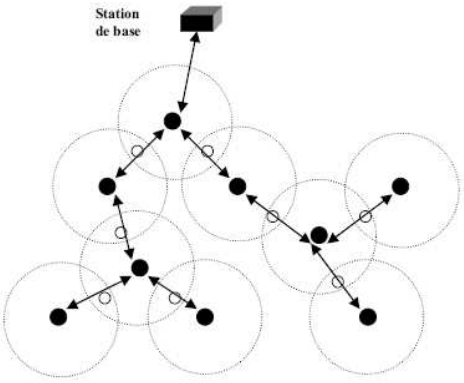

Figure 2.10 – Topologie résultant du protocole TopDisc

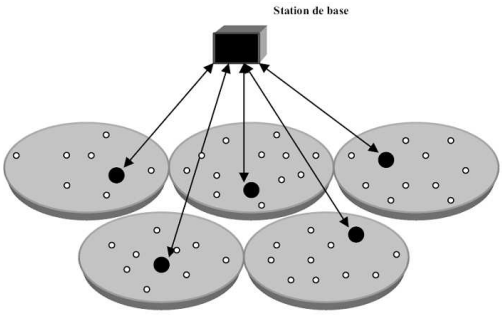

Figure 2.9 – Topologie résultant du protocole LEACH

La seconde méthode que l'on peut rencontrer, est nommée TopDisc [Deb 01]. Contrairement à LEACH, TopDisc s'appuie sur une structuration à plusieurs niveaux hiérarchiques ce qui lui permet d'élargir son champ d'action. Cependant, les données des noeuds sont centralisées avant d'être traité par le superviseur pour déterminer les différents rôles. Pour cette méthode, le rôle "passerelle" est ajouté ce qui permet de définir des noeuds de liaison entre les différents clusters. La Figure 2.10 représente la topologie résultant en appliquant cette méthode sur un réseau. L'avantage de cette méthode est de permettre d'organiser des réseaux vastes qui ne sont pas directement visibles par le superviseur. Par contre, la gestion centralisée des rôles implique une augmentation du volume de données échangées ce qui dans un RCSF est très pénalisant d'un point de vue énergétique. On réservera donc TopDisc pour des réseaux avec peu de noeuds afin de limiter ses échanges.

Enfin, la formation et la gestion des clusters peuvent être effectuées automatiquement en utilisant des critères simples, comme le précise la méthode LiveA [Diao 07]. Cette méthode s'appuie pour former les clusters sur deux critères simples qui sont modélisés par les formules suivantes :

$$\begin{gathered} \forall (N_{M1}, N_{M2}), P < D_M < 2P \\ \text{et} \\ \exists i \mid N_i \in (V_{M1} \bigcap V_{M2}) \end{gathered} \tag{2.1}$$

avec N_{Mi} le noeud maître i, N_i le noeud i, D_M la distance entre deux noeuds maître, P la portée d'un noeud, et V_{Mi} l'ensemble des voisins du noeud maître i.

Le premier critère définit l'élection des noeuds maîtres en fonction des liaisons qu'ils peuvent avoir avec leurs voisins, tandis que le second définit le nombre de liaisons communes entre chaque noeud maître. Pour améliorer les chemins de communication, ce dernier critère peut évoluer afin de définir un nombre minimal de noeuds communs entre cluster. On aura donc n chemins pour transférer des données d'un cluster à un autre jusqu'au superviseur. L'avantage de cette méthode est de permettre un fonctionnement aussi bien centralisé que décentralisé. Dans le deuxième cas, l'utilisateur ne définit que les contraintes (liens entre cluster, portée...) puis le réseau applique l'algorithme précédent pour s'organiser. Le principal inconvénient de cette méthode est le volume de messages nécessaires pour effectuer l'organisation car chaque noeud doit connaître les voisins de ses voisins. Ensuite, le mécanisme de définition des rôles demande un surplus de messages qui pour de gros réseaux peut nuire ou faire tomber la liaison par surcharge de la bande passante.

Bilan des méthodes d'auto-configuration

En définitive, l'organisation d'un RCSF constitue un atout important dans la lutte contre la consommation énergétique liée aux communications sans fil. En guise de comparaison entre ces 3 méthodes, la Table 2.4 rappelle pour chacune d'elles leurs caractéristiques. Avec cette table, on constate alors que ces méthodes sont assez proches les unes des autres et demandent une station de base plus puissante pour traiter les informations reçues des noeuds pour en déduire une topologie. Ce sera donc un point sensible car elle devra en plus d'appliquer l'auto-configuration, gérer les collisions et le goulot d'étranglement qu'elle introduit. Contrairement au mode centralisé, un mode décentralisé comme celui de LiveA permet d'éviter cela. En contre partie, la quantité de messages échangés devient plus importante.

En définitive, la méthode d'auto-configuration idéale sera en mode décentralisé avec peu de messages échangés, et intégrera une méthode d'équilibrage de l'énergie en faisant évoluer la topologie périodiquement. Le réseau sera découpé en clusters communicants entre eux par l'intermédiaire de noeuds relais formant ainsi un maillage du réseau.

2.3.2 Routage des messages

Le routage des informations dans un RCSF est un sujet de recherche très important sur lequel encore aujourd'hui il n'y a pas de réponses universelles. Un classement des différents protocoles a été proposé en fonction de leur type proactif, réactif ou hybride [Jiang 04] mais aussi en fonction de leur mode de fonctionnement [Al-Karaki 04], à plat, hiérarchique, localisé ou améliorant la QoS. Ceux définis comme proactif, tiennent à jour périodiquement la table de routage en envoyant des messages pour vérifier les liens tandis que les protocoles réactifs ne le font qu'au moment de l'envoi d'un message. Les protocoles de routage multi-saut à plat utilise des noeuds identiques managés directement par la station de base. Les requêtes sont alors envoyées vers une zone ciblée du réseau car le nombre important de noeuds empêcherait leurs bons

Table 2.4 – Comparatif des méthodes d'auto-configuration

	LEACH	TopDisc	LiveA
Type d'architecture	centralisé	centralisé	centralisé ou décentralisé
Architecture d'administration	un niveau de clusters	hiérarchie de clusters	hiérarchie de clusters
Orientation	routage	administration de réseau	administration de réseau
Construction de clusters	équations	équations	équations
Critères utilisés	énergie disponible des noeuds	position géographique des noeuds	voisinage des noeuds
Chevauchement des clusters	aucun	avec ou sans	avec
Auto-configuration	immédiatement après le déploiement	sur demande de la station de base	immédiatement après le déploiement
Election des noeuds maîtres	rotation des noeuds maîtres	Noeuds maîtres fixes	Noeuds maîtres fixes
Evolution	dynamique	statique	statique
Volume de données	variable (rotation périodique des noeuds maîtres)	contrôlé par le superviseur	importants en mode décentralisé

traitements. L'exemple typique de cette catégorie est le protocole Sensor Protocol for Information via Negotiation (SPIN) [Kulik 02] qui implémente des méta-données sur les données afin de ne les transmettre qu'aux noeuds les demandant réellement. Le routage hiérarchique quant à lui, divise le réseau en zones ou clusters administrés localement par un noeud disposant de traitements spécifiques. Dans cette catégorie, on retrouve le protocole LEACH [Heinzelman 00] que l'on présentera par la suite. Ensuite, les protocoles dits localisés [Stojmenovic 05] s'appuient sur les positions géographiques des noeuds pour évaluer le meilleur trajet pour un message. Le protocole CIVIC présenté par la suite entre dans cette catégorie puisqu'il utilise la position GPS des noeuds pour diffuser les messages. Enfin la dernière catégorie vise à améliorer la QoS dans les RCSFs. Les noeuds n'ont plus seulement la contrainte énergétique à prendre en compte, la qualité de réception vient s'y ajouter. Ainsi, ils devront obtenir un compromis entre ces deux contraintes pour diffuser les données. Afin de réaliser la fonction de routage, de nombreux protocoles dont Optimized Link State Routing Protocol (OLSR) [Clausen 03], Ad hoc On-Demand Distance Vector (AODV) [Perkins 03], ou Communication Inter Véhicules Intelligente et Coopérative (CIVIC) [Chanet 07, Diao 08] ont été réalisés pour améliorer la diffusion des données dans un RCSF.

Le protocole OLSR est une optimisation de l'algorithme classique d'états des liens adaptée aux exigences d'un RCSF. Le concept clé utilisé dans ce protocole est celui des relais 'multipoint' (MPR). Les MPR sont des noeuds sélectionnés qui transmettent les messages broadcastés au cours du processus d'inondation. Cette technique réduit considérablement les messages par rapport à un mécanisme classique d'inondation, où chaque noeud retransmet chaque message quand il en reçoit la première copie. Dans OLSR, l'information d'état des liens est générée seulement par des noeuds élus MPR. Ainsi, une seconde optimisation est réalisée en minimisant le nombre de messages de contrôle diffusés dans le réseau. Comme troisième optimisation, un MPR peut choisir de signaler seulement ses liens entre lui et ses MPR parents. Par conséquent, contrairement à l'algorithme classique d'état des liens, des informations partielles d'état des liens sont distribuées dans le réseau. Cette information est ensuite utilisée pour le calcul d'itinéraire. OLSR fournit des routes optimales en termes de nombre de sauts.

AODV est un protocole destiné à être utilisé par des noeuds mobiles dans un réseau ad hoc. Il a été défini pour être utilisé avec des faibles ressources processeur et mémoire et il s'adapte rapidement aux changements des liens. La majeure partie du temps, AODV ne produit pas de trafic supplémentaire dans le réseau, il est défini comme réactif, c'est-à-dire qu'il échangera des messages de routage que lorsque cela sera nécessaire.

CIVIC est un protocole de routage directionnel hybride développé, dans un premier temps, pour des applications inter véhicules telles que la gestion d'une flotte de véhicule. Peu à peu, il a évolué pour traiter la problématique des réseaux agri-environnementaux. CIVIC est implémenté pour être utilisé au niveau de la couche Réseau de la pile OSI. En ce qui concerne son mode de fonctionnement, celui-ci met en concurrence les différents noeuds présents dans un cône défini par rapport aux positions géographiques des noeuds diffuseur et récepteur du message. Chaque noeud présent est en concurrence en fonction de ces ressources disponibles pour faire suivre le message.

Les protocoles de routage présentés sont des protocoles autonomes pouvant être employé sur

la majorité des médiums et contextes logiciels. D'autres sont liés ou intégrés dans une pile de protocole comme IPv6 Routing Protocol for Low power and Lossy Networks (RPL) [Winter 09], Dynamic MANET On-demand for 6LoWPAN Routing (DYMO-low) [Kim 07a], 6LoWPAN Ad Hoc On-Demand Distance Vector Routing (LOAD) [Kim 07b], ou Hierarchical Routing over 6LoWPAN (HiLow) [Kim 07c], inclus dans la pile de protocoles IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) [Montenegro 07].

La pile de protocoles 6LoWPAN a été créée pour répondre au besoin de communiquer au sein des réseaux sans fil à faible consommation, Low-power Wireless Personal Area Network (LoWPAN). L'objectif premier [Montenegro 07] est de fournir la connectivité IPv6 sur des réseaux ne disposant que de très peu de ressources processeur/mémoire dont les échanges sont restreints à quelques octets sans fragmentation des messages par le médium. Pour cela, la norme [Montenegro 07] prévoit une redéfinition des paramètres échangés dans les paquets pour en réduire leur taille. Ainsi, 6LoWPAN applique une compression permettant de réduire les octets envoyés en utilisant tous les bits disponibles et en codant différemment les paramètres de l'entête IPv6. Par ailleurs, pour effectuer le routage dans ce type de réseaux, le protocole RPL a été développé pour fournir cette fonctionnalité sur des réseaux utilisant la pile 6LoWPAN. A la question de savoir pourquoi utiliser le protocole IPv6 sur ce type de réseau, la réponse est assez simple. La migration vers cette connectivité de tous les réseaux mondiaux permettra d'adresser un nombre bien plus important de machines dans le monde. Devant une telle quantité d'adresses, les RCSFs voient une nouvelle opportunité de se développer. Pour cela, l'idée dans le futur sera de rendre ces réseaux accessibles dans le monde entier afin de permettre à n'importe quel utilisateur de suivre l'évolution d'un phénomène (pollution dans une ville, des eaux, etc.) ou d'un événement (trafic routier, manifestations culturelles, etc.). En rendant compatible les mondes des RCSFs et des ordinateurs personnels, la migration et leur fusion se feront très facilement et à faible coût.

Ces différents protocoles sont utilisés comme base de diffusion des données dans le réseau. La solution d'administration pourra choisir de reprendre ou redéfinir certains paramètres de son point de vue. Le nombre de sauts d'un message d'administration pourra, par exemple, définir le nombre de clusters concernés ou le nombre de noeuds devant traiter le message. L'adressage des noeuds se fera, par contre, directement par le protocole de routage qui est plus adapté et surtout qui est à une couche identique ou plus basse que l'administration. Les messages d'administration seront encapsulés dans le message généré par le routage. On voit donc que ces protocoles sont nécessaires pour accéder le plus rapidement aux noeuds du réseau en minimisant l'impact énergétique des transmissions sur ceux-ci.

2.3.3 Compression des données

Sur un noeud, le gros consommateur d'énergie étant le médium de communication, il convient alors de réduire son utilisation globale, le volume de données échangées ainsi que celles sauvegardées sur le noeud. Pour cela, il est courant de trouver des techniques de compression de données pour minimiser les temps d'utilisation du médium de communication ou pour minimiser les données sauvegardées dans la mémoire du noeud. Par définition, cette opé-

Table 2.5 – Comparatif des méthodes de compression

Codage	Performances	Compression (Décompression)	Ressources utiles	Diffusion du dictionnaire
RLE	Moyenne	Simple (Simple)	Faible	Non
Huffman statique	Bonne	Complexe (Simple)	Haute	Non
Huffman semi-adaptatif	Bonne	Complexe (Simple)	Haute	1 par groupe de message
Huffman adaptatif	Très Bonne	Complexe (Simple)	Haute	1 à chaque message
LZW	Bonne	Normale (Normale)	Moyenne	Non

ration consiste à transformer une suite de bits A en une suite de bits B plus courte contenant les mêmes informations, en utilisant un algorithme particulier. Elle peut être avec ou sans perte suivant l'importance des données et la tolérance sur celles-ci.

Dans les RCSFs, tous les algorithmes présents dans le paysage informatique ne conviennent pas aux contraintes liées à ce domaine d'application. Certains comme Run-Length Encoding (RLE), Huffman [Huffman 52], ou Lempel-Ziv-Welch (LZW) [Welch 84] ont été adaptés aux RCSFs et utilisés pour transmettre des données. L'algorithme RLE utilise la fréquence des caractères pour compresser les données. Une suite de 'n' caractères identiques est remplacée par le couple (nombre d'occurrences, caractère). L'algorithme de Huffman quant à lui, modifie le codage des caractères en fonction du nombre d'occurrences dans le fichier source. Le codage du caractère le plus fréquent utilisera peu de bits tandis que le moins fréquent utilisera plus de bits pour le représenter. Pour la décompression, il est nécessaire de disposer du dictionnaire des caractères. Enfin, l'algorithme LZW forme un codage adaptatif qui prend en compte la répétabilité de certaines suites de caractères dans un texte. Pour chaque nouvelle suite de caractères, un nouveau code est créé. La décompression est autonome, c'est-à-dire qu'elle n'a besoin de rien d'autre que le fichier compressé pour reformer le fichier source. La Table 2.5 présente les différentes caractéristiques de ces algorithmes de compression.

Avec un taux de compression plus important, l'algorithme de Huffman est le plus performant et le plus légitime dans les RCSFs. Cependant, il est nécessaire d'échanger un dictionnaire de caractères pour pouvoir effectuer la décompression ce qui augmente le risque de pertes ou d'erreurs sur le noeud. L'algorithme LZW est celui qui offre de bonnes performances sans diffuser des données supplémentaires. La création de nouveaux codes pour les séquences de caractères impose alors d'augmenter le nombre de bits utiles pour représenter chaque caractère.

2.3.4 Synthèse générale

Nous avons pu voir différentes méthodes pour organiser un réseau, diffuser et réduire les données échangées. Ces méthodes bien qu'externes à l'administration contribuent au bon fonctionnement des échanges entres les noeuds. Choisir de découper un ensemble de noeuds en clusters par exemple, contribuera, dans un premier temps, à la préservation de l'énergie, et, dans un second temps, à améliorer les transmissions. Ensuite en améliorant les conditions de diffusion d'un point A vers un point B via des protocoles de routage, on optimisera les temps pour accéder au noeud et pour y effectuer une opération. Ces méthodes invisibles pour l'administration définissent un socle stable pour diffuser les directives d'administration vers les noeuds. Sans ceci, l'accès à un noeud consommerait énormément d'énergie car le message concernant un noeud serait diffusé sur tout le réseau même vers les noeuds les plus éloignés du destinataire. Par ailleurs, l'administration peut devenir gourmande en informations stockées et diffusées. La compression vient ici en appui pour limiter ces volumes et minimiser ainsi l'impact de l'administration sur le réseau. Il est important de rappeler que le coût énergétique induit par la compression reste très inférieur au coût de diffusion d'une donnée. Il est donc intéressant de prendre un peu de temps avant d'envoyer les données vers un noeud. De plus, la mise à jour à distance demande de diffuser d'importants volumes de données. La compression participe alors à la réduction du fichier source afin de réduire les temps de mise à jour par exemple.

Pour résumer, l'administration s'appuie sur une base solide de méthodes permettant de diffuser et minimiser les transmissions de données. Le poste 'communication' coûte le plus en termes d'énergie ce qui oblige l'administration à le prendre en compte afin de limiter ses échanges. Toutefois, les solutions d'administration doivent être indépendantes de ces techniques afin d'avoir une adaptabilité quelque soit la base utilisée. L'utilisateur doit pouvoir choisir les méthodes qu'il souhaite pour diffuser ou organiser. Par ailleurs, le découpage du réseau en plusieurs zones administrables indépendantes permet de définir des traitements plus spécifiques en fonction des contraintes subies. De plus, certaines tâches d'administration ne demandent pas de faire appel au superviseur global car le manager local peut les réaliser.

2.4 Conclusion

Les RCSFs de part leur vaste champ d'applications, subissent de nombreuses contraintes telles que l'énergie, les ressources matérielles et la communication. Les besoins motivés par ces contraintes trouvent des réponses, mais aucune n'est générique. Chaque méthode développée est bien souvent orientée vers une application ou une contrainte spécifique dans un environnement tout aussi spécifique. Il n'y a pas à ce jour une réponse générique à toutes les contraintes. La problématique de l'administration de RCSF reste entière tant le sujet est vaste. Répondre à tous les besoins des noeuds, en termes de gestion, demande de développer de nombreux mécanismes pour gérer l'énergie, les ressources matérielles et les communications. De ce point de vue, les méthodes développées aux couches inférieures participeront nécessairement à la préservation de ces ressources. Par contre, pour adapter le noeud aux contraintes qu'il subit une solution d'administration est obligatoire.

Sans administration, il est difficile de déployer des RCSFs sur des applications vastes ou à risques. Les solutions présentées proposent de nombreux mécanismes répondant à quelques contraintes mais aucune ne gère la totalité des contraintes. Les solutions d'administration ne permettent pas encore à ce jour de produire une réponse générique à la problématique de supervision des OICs. Les solutions basées sur le protocole SNMP ne proposent qu'une adaptation de l'agent pour pouvoir l'embarquer sur les noeuds. La diversité des informations administrées est alors faible en se cantonnant à la problématique de la gestion de la mobilité des noeuds et de la stabilité des liens. Une approche centralisée a été évoquée mais le faible degré d'évolution réduit son utilisation sur les réseaux agri-environnementaux par exemple. Par ailleurs, les autres approches basées sur une couche logicielle intermédiaire, middleware, proposent des mécanismes intéressants pour l'administration comme la programmation modulaire ou le système de mise à jour. Cependant, ces méthodes demandent l'ajout de traitements sur les noeuds et donc l'augmentation des performances des noeuds pour obtenir la même application que sans cette technique. Les commandes seront interprétées à la réception par cette interface intermédiaire avant d'exécuter les actions qui leur sont associées. Enfin, il existe des méthodes spécifiques à une contrainte comme la gestion d'énergie. Ces méthodes sont bien souvent liées à une architecture matérielle et logicielle les rendant difficile à mettre en oeuvre dans un contexte différent. Néanmoins, les techniques utilisées pour limiter l'utilisation des noeuds et pour préserver l'énergie constituent des atouts intéressants qu'une solution d'administration doit prendre en compte.

En parallèle à la supervision du réseau, des méthodes complémentaires doivent être mises en place pour traiter les points associés à la diffusion des données dans le réseau. L'organisation du réseau est alors un point clé pour faciliter les échanges mais aussi pour limiter l'impact de la diffusion des messages sur les noeuds. La piste retenue par tous est la clusterisation hiérarchique afin d'atteindre n'importe quel noeud du réseau sans faire appel aux noeuds inutiles pour la diffusion. Cette technique de découpage est présente au niveau du routage mais aussi au niveau de l'administration. Pour cette dernière, le découpage du réseau consiste à définir le rôle des noeuds afin de répartir les fonctions d'administration. Les deux découpages peuvent alors être différents. Des noeuds reliés directement d'un point de vue 'administration' ne le sont pas

obligatoirement d'un point de vue 'routage'. La diffusion des données se fait donc à partir d'un protocole de routage adapté aux contraintes du réseau. Qu'il soit proactif, réactif, ou hybride, leur rôle reste la découverte de la route optimale entre une source et une destination. L'administration se base sur ces méthodes pour diffuser ces messages et requêtes. La compression des données d'administration devient alors intéressante pour limiter les échanges notamment pour la mise à jour à distance des noeuds.

Par ailleurs, pour l'ensemble de ces solutions d'administration, les paramètres sont collectés à la demande, c'est-à-dire qu'il y aura un message diffusé par paramètre souhaité. L'aspect intrusif de ces solutions influencera le fonctionnement du noeud, son autonomie et ses échanges dans le réseau. Une limitation des accès aux données doit être envisagée à travers des techniques de déduction et de connaissances du contexte. Par exemple, certaines informations comme l'état d'un capteur peuvent être déduites des données qu'il envoie.

En définitive, les solutions d'administration ne permettent pas encore à ce jour de produire une réponse générique à la problématique de gestion des RCSFs. Même si de nombreuses idées sont très innovantes, aucune ne peut être migrée ou utilisée directement sur n'importe quelle plateforme matérielle dans n'importe quel environnement. La problématique énergétique a été prise en compte dans les solutions d'administration présentées à travers un contrôle des données diffusées, du temps de fonctionnement des noeuds ou de leurs organisations. Cependant, les solutions produites ne parviennent pas à répondre à l'ensemble des caractéristiques d'un outil de management. C'est pourquoi, le chapitre suivant présentera une méthode d'administration appelée LiveNCM qui tente de répondre à toutes ces caractéristiques tout en étant la moins intrusive possible et en adoptant certaines idées vues ici.

$\sim \star \sim \star \sim \star \sim$

Chapitre 3

Une nouvelle approche de l'administration

Le chapitre précédent présentait différentes solutions d'administration existantes pour les RCSFs. Certaines offrent des fonctionnalités intéressantes permettant une meilleure gestion des ressources ou une meilleure modularité, mais aucune ne correspond entièrement aux demandes d'administration des RCSFs. Il est donc apparu nécessaire de développer une nouvelle solution de supervision de réseau d'OICs afin de traiter plus efficacement les contraintes actuelles des RCSFs. En complément, nous choisirons d'introduire des restrictions supplémentaires, afin de limiter les volumes de données échangées au sein des réseaux pour minimiser l'impact de l'administration sur la source énergétique et sur l'application.

Ce chapitre est consacré à la présentation de l'architecture d'administration LiveNode Non invasive Context-aware and Modular management (LiveNCM) et de son implémentation au sein d'un RCSF. Les contraintes générales d'un noeud seront directement prises en compte au cours du développement afin fournir une solution adaptée aux problèmes liés à chaque noeud. La dimension générique de LiveNCM imposera d'effectuer des choix logiciels certes moins performants mais permettant de faciliter son utilisation pour les échanges d'informations entre les noeuds et l'utilisateur tout en optimisant sa portabilité.

3.1 LiveNCM : LiveNode Non invasive Context-aware and Modular management

Le domaine des RCSFs, comme nous avons pu le constater dans le chapitre précédent, contient encore de nombreuses problématiques non résolues. Les solutions existantes d'administration de RCSF proposent une multitude d'idées intéressantes offrant une bonne modularité des applications embarquées ainsi qu'une meilleure gestion des ressources. Si une architecture modulaire est utile pour la mise à jour des applications ou des progiciels des noeuds, la charge engendrée pour traiter et mettre en oeuvre les nouvelles versions peut devenir lourde. De même, les solutions optimisant les ressources s'appuient souvent sur des systèmes d'exploitation spécifiques (TinyOS, MagnetOS, SenOS, etc.) ce qui réduit leur interopérabilité. Au regard de

ces différents points, l'adaptation de l'une des méthodes présentées dans le chapitre précédent demande un travail important d'adaptation aussi bien au niveau de son développement qu'au niveau de son environnement de travail. De plus, de nouvelles contraintes devront être ajoutées et porteront sur le volume des données échangées ainsi que sur la réactivité des noeuds. Au final, une nouvelle méthode d'administration est nécessaire pour intégrer tous ces points en se basant sur les différents avantages des solutions existantes.

En se basant sur ce constat et sur les différents critères cités à la section 2.1.1, la solution LiveNCM a pu voir le jour. L'objectif de cette technique est bien sûr de faciliter l'accès aux données par l'utilisateur en rendant les échanges totalement transparents pour celui-ci et le moins intrusif possible pour le réseau. Pour limiter l'impact de l'administration sur l'application et plus généralement sur le réseau, LiveNCM est conçu pour permettre une réduction des échanges concernant la collecte des données administratives. Cela passe par l'intégration du concept de diagnostic indirect et des techniques d'estimation et de compression. Le principe est de déduire des informations d'état en fonction des différents messages échangés. Si dans les méthodes présentées précédemment, l'autonomie des noeuds était déjà un élément important à prendre en compte, ici, on associe ces méthodes à un nouveau modèle d'administration qui permettra d'amplifier leurs gains énergétiques. Chaque message transitant devient un message d'administration dans lequel il est toujours possible de déduire des informations sur l'état de l'émetteur.

Avec une diffusion importante et de nombreux outils d'administration, il semble alors intéressant de s'appuyer sur le protocole SNMP, présenté comme le protocole d'administration des réseaux filaires, afin de centraliser et d'organiser les échanges avec les RCSFs. Malgré des messages de taille conséquente (100 à 200 octets), ce protocole a atteint une notoriété importante dans les réseaux filaires ce qui en fait l'acteur majeur des solutions d'administration. De plus, de nombreux outils ont vu le jour, et maintenant on trouve très facilement dans tous les langages de programmation des fonctionnalités SNMP. Pour ces raisons, LiveNCM a choisi d'adapter ce protocole pour avoir une interface connue et reconnue entre les noeuds et l'utilisateur. La liaison entre ce protocole et les noeuds nécessitera l'utilisation d'une interface de communication transférant les données d'un acteur à un autre. Cette interface a pour objectifs de permettre un accès facile aux noeuds et d'intégrer quelques fonctionnalités périodiques d'administration comme la synchronisation. En externalisant cette partie, il est alors possible de déporter sur des sites différents l'agent et la passerelle. Par ailleurs, le protocole SNMP s'appuie sur le mécanisme UDP/IP pour envoyer les requêtes ce qui permet d'y avoir accès via n'importe quel réseau local filaire du type Local Area Network (LAN) (réseau informatique local) ou Wide Area Network (WAN) (réseau informatique étendu). La passerelle de communication pourra donc tirer profit de ceci pour communiquer avec l'agent. Les échanges dans cette architecture se baseront sur un protocole de communication autre que SNMP pour prendre en compte les contraintes de communication dans les RCSFs ainsi que la contrainte de taille des messages. Celui-ci pourra reprendre des caractéristiques des existants comme 6LoWPAN, notamment pour la formalisation des données, et utiliser différents protocoles de routage pour transmettre les requêtes. Par ailleurs, afin de rendre générique cette solution d'administration et surtout le format des messages, il sera intéressant d'inclure différents types de commandes pour différents

traitements. On peut penser alors à l'utiliser à tout niveau de la communication, que ce soit pour l'administration ou pour les collectes de données.

D'un point de vue global, la solution LiveNCM propose une nouvelle structure d'administration basée sur un seul agent SNMP adapté centralisant les remontées d'informations des noeuds via une passerelle de communication dédiée et indépendante. La Figure 3.1 présente une première vision de l'architecture recherchée par laquelle l'utilisateur dialoguera avec les noeuds.

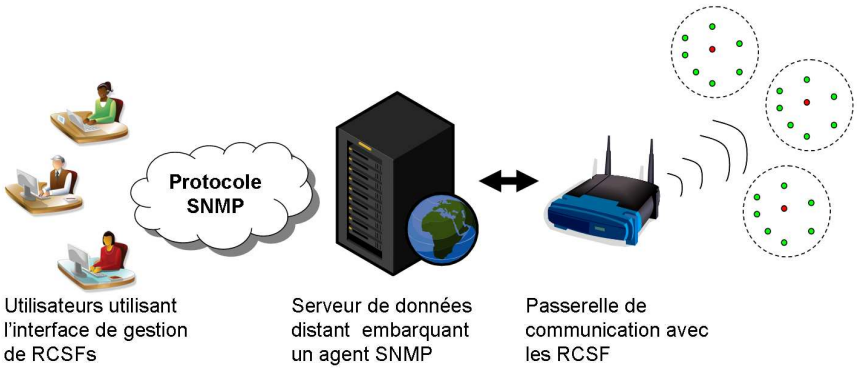

Figure 3.1 – Architecture d'administration souhaitée par LiveNCM

L'extension de l'agent SNMP, présent sur la machine centralisant les informations, se fera sous forme d'un sous-agent qui intégrera l'ensemble des nouvelles données à superviser et répondra aussi bien aux requêtes de l'utilisateur qu'à celles des noeuds. Les différents éléments de la solution reprennent une partie des caractéristiques des méthodes existantes tout en introduisant de nouvelles contraintes sur les échanges et sur l'administration. L'utilisateur pourra mettre en oeuvre n'importe quel outil d'administration SNMP disponible pour manager à distance un ensemble de RCSFs constitués de plusieurs noeuds interfaçant plusieurs capteurs. La dynamique de ces réseaux implique alors d'avoir une solution d'administration réactive et adaptable pour prendre en compte le moindre changement de chaque élément. Cela se traduit donc par une nouvelle contrainte sur la réalisation du sous-agent SNMP qui devra permettre, à tout instant, des modifications de la structure de données utilisée. Ce mémoire présente alors l'ensemble des étapes du développement de la solution d'administration LiveNCM et plus particulièrement la partie de dialogue avec les noeuds. Nous travaillerons alors sur le sous-agent SNMP, la passerelle de communication, l'agent embarqué sur les noeuds ainsi que sur le protocole de communication utilisé. La section suivante présentera, en détails, ces différents éléments en s'attardant sur la réponse apportée aux nouvelles contraintes de dynamisme, de volume de données échangées, et de réactivité.

3.2 Description détaillée de LiveNCM

L'étude des différentes solutions d'administration dans le chapitre précédent nous amène à considérer plusieurs types d'administration axés vers les données, les noeuds ou les deux. Par

ailleurs, les méthodes d'organisation peuvent faire partie intégrantes de la gestion de réseau. Dans un premier temps, nous nous attarderons sur une administration axée vers les noeuds sans prendre en compte la gestion des données. Ce point sera plutôt traité par des méthodes implémentant un système de fichiers (par exemple LiveFile [De Sousa 07]) ou une base de données distribuée (par exemple COUGAR [Yao 02]). Toutefois, un minimum de gestion de données sera à intégrer pour sauvegarder et maintenir à jour l'état courant du noeud ainsi que certains paramètres essentiels. L'organisation du réseau se fera indépendamment de la solution LiveNCM afin de pouvoir adapter la majorité des méthodes existantes à l'architecture d'administration choisie.

3.2.1 Méthodes de réduction des données

Réduction des échanges via la méthode de diagnostic indirect

LiveNCM décrit un ensemble de concepts et de techniques permettant un accès distant aux données d'un noeud. L'aspect intrusif des méthodes existantes ne permet pas de préserver une autonomie maximale du noeud. Pour chaque information à collecter, une requête doit être formée et envoyée. Avec des réseaux de plusieurs milliers de noeuds, la collecte de l'état de chaque noeud peut engendrer l'effondrement d'un réseau et/ou nuire au fonctionnement de l'application embarquée. Ce problème obtient une réponse dans LiveNCM par l'intermédiaire de la notion de diagnostic indirect [Jacquot 09]. Cette technique innovante permet de déduire des informations d'un élément en se basant, en partie, sur les échanges de données. Dans le contexte des applications agrienvironnementales, en prenant en compte le contexte de l'environnement de l'application par exemple (saisons, cycle jour/nuit, modèles, etc.), il est courant de pouvoir déduire des informations à partir du dernier message reçu qu'il soit d'administration ou de données. L'exemple de la Figure 3.2 met en évidence quelques informations qui peuvent être déduites lors de la réception d'un message daté de collecte de données. Les erreurs de synchronisation seront détectées par comparaison avec l'horloge du récepteur entre deux messages reçus. L'information énergétique est déduite de l'entête des messages reçus. En effet, un champ autorisant une évaluation très simple du niveau énergétique pourra être inclus dans chaque message. Enfin, la qualité, la calibration et la validité des mesures seront déduites en prenant en compte l'historique des mesures, les plages de valeurs, ou alors les valeurs transmises par les noeuds voisins. Dans le cas des applications agri environnementales, des noeuds peuvent disposer de plusieurs capteurs qui seront déployés assez proches les uns des autres. Il est alors possible, pour la plupart des grandeurs observables, de mettre en relation deux capteurs afin de détecter les erreurs de mesure ou de calibration (mesures abérantes).

A partir d'un seul message, le diagnostic indirect permet de déduire au moins 4 informations concernant l'état d'un noeud émetteur (erreur de synchronisation, niveau de batterie, qualité et fiabilité de la chaîne de mesure). Tous les messages échangés sont alors considérés comme une source d'information pour l'administration, et passent automatiquement par LiveNCM avant d'atteindre leur destination. La réactivité du noeud est alors légèrement affectée par des traitements simples de vérification. Au niveau du superviseur, le traitement de tous les messages reçus n'influencera que très peu sa réactivité car il dispose de ressources importantes

permettant d'appliquer plus rapidement les différentes fonctions d'administration.

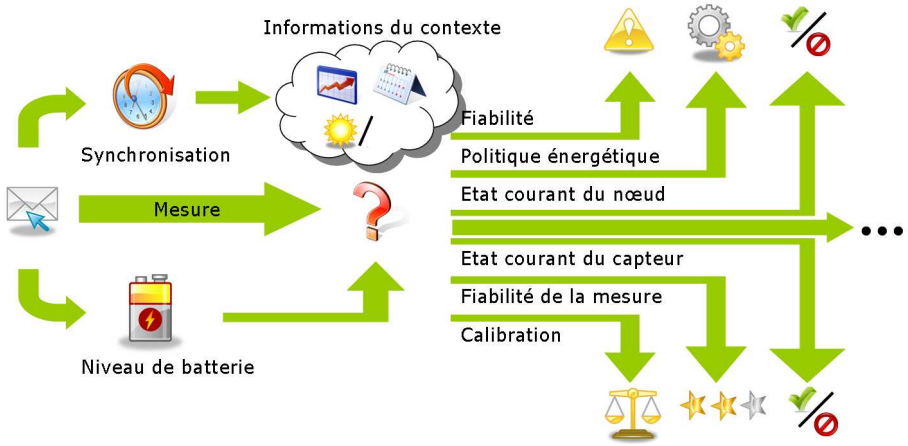

Figure 3.2 – Exemple de diagnostic à partir d'un seul message

Si l'on considère qu'à partir d'un seul message, il est possible de déduire au moins 4 informations, le superviseur ne demandera ces informations plus qu'à titre de confirmation. Ce sera aux noeuds, suivant les critères d'administration définis, de prendre la décision d'informer le superviseur d'un paramètre important ou ayant changé. Ces critères peuvent aller du silence jusqu'à l'envoi systématique du paramètre ayant évolué. A des niveaux intermédiaires, les informations déduites par le diagnostic indirect permettront de compléter les remontées d'informations des noeuds. Dans ce cas, le nombre de messages formés pour obtenir les mêmes informations est divisé par 4, ce qui réduit nécessairement l'intrusion de l'administration dans le réseau et sur les noeuds. Sans compression ni aucune autre méthode de réduction de données, le diagnostic indirect constitue une nouvelle approche innovante limitant les échanges entre les noeuds. Il pourra être appliqué aussi par les noeuds pour prédire par exemple la défaillance d'un voisin ou pour collaborer et corriger mutuellement. Dans un RCSF, la collaboration des noeuds peut permettre une amélioration du fonctionnement général. Pour des capteurs proches et une grandeur évoluant de manière identique, il est tout à fait possible, à travers le diagnostic indirect, qu'un noeud se considérant alors comme valide devienne invalide car il fournit des valeurs aberrantes par rapport à ses voisins proches.

Cette technique est à la base de la solution d'administration LiveNCM sur la problématique de réduction des données échangées. En complément, il est possible d'intégrer un estimateur simple par exemple afin de contrôler le flux de données pour des grandeurs évoluant lentement et de prédire, entre autres, des états futurs de batterie.

Réduction des échanges via des estimateurs

Afin de réduire les données échangées, les techniques de compression classiques tiennent une place très importante. Les échanges sont compressés sans perte mais la fréquence d'envoi des messages reste la même. Avec un volume de données diminué, l'impact énergétique des

communications sur la source énergétique est minimisé. Cependant, pour certains relevés comme la température, la pression ou l'humidité par exemple, une technique basée sur un estimateur peut présenter de meilleures performances qu'un algorithme de compression. Si déjà, on envoyait moins de données en compressant, avec un estimateur, il est possible d'éliminer les données inutiles en introduisant une marge d'erreur. Le gain d'une telle technique se traduit par une forte diminution des transferts de données. C'est pour cette raison que nous avons choisi d'étudier et d'intégrer un estimateur dans la solution d'administration LiveNCM [Jacquot 08]. Toutefois, le choix de l'estimateur reste entier. En effet, il devra permettre une bonne estimation, en utilisant un minimum de ressources processeur et mémoire tout en ne nécessitant que très peu d'échanges pour sa maintenance. Pour réaliser cette opération, différents algorithmes linéaires et non linéaires sont possible, mais nous avons opté pour des techniques polynomiales simples et peu coûteuses en opérations de mise à jour. Elles sont donc basées sur des polynômes d'ordre supérieur ou égal à 0 pour lesquelles les calculs de mises à jour sont rapides et simples. Un estimateur s'appuiera sur un seuil fixé par l'utilisateur pour détecter les échantillons à envoyer. La tolérance ou marge d'erreur du signal représente alors la zone dans laquelle peut évoluer le signal estimé sans nécessité l'envoi et la mise à jour du modèle. Enfin, la résolution d'un signal correspond à la précision de la mesure en fonction du capteur et de la chaîne d'acquisition employés.

Les estimateurs basés sur des polynômes permettent en fonction de leur ordre de s'approcher au mieux de la courbe observée. Plus l'ordre est élevé, plus le signal estimé peut être précis. En contrepartie, la précision dépend du nombre d'échantillons utilisés pour former le polynôme mais aussi de la complexité des calculs engendrés par la mise à jour du modèle. C'est pourquoi, nous avons choisi d'étudier cette technique jusqu'à l'ordre 3. Un polynôme d'ordre 0 revient à discrétiser le signal observé en échelon, c'est-à-dire que l'on garde la même mesure tant que l'on ne dépasse pas le seuil de détection. A cet ordre, la tendance de la courbe n'est pas préservée et la précision sera moindre. Un polynôme d'ordre 1 n'utilisera quant à lui peu de ressources de calcul pour mettre à jour son modèle. Avec seulement 2 points, il est possible de redéfinir le polynôme en seulement 7 opérations simples (affectation, addition, division). A chaque modification du modèle, il n'est nécessaire d'envoyer que le dernier échantillon, et ensuite, on se basera sur le précédent échantillon reçu. La Figure 3.3 présente un exemple d'application d'un estimateur d'ordre 1. Les droites Di représentent les différents modèles d'estimation appliqués sur la courbe relevée (A). La courbe (B) représente alors le résultat obtenu. De plus, un tel modèle aura la capacité de s'adapter à l'évolution des données en choisissant d'envoyer plus de points dans des zones critiques comme le sommet de l'arche d'une sinusoïde par exemple. On peut donc le qualifier d'estimateur simple et adaptatif.

Ensuite, pour les polynômes d'ordres supérieurs (2 et 3), il est nécessaire de disposer de plus de points pour permettre la construction du modèle (3 pour le second ordre et 4 pour le troisième ordre). Les calculs des différents coefficients de ces deux modules engendrent plus d'une centaine d'opérations à chaque mise à jour du modèle. Les temps de calcul sont alors plus longs, et il est nécessaire de transmettre, à chaque fois, tous les points participants à la mise à jour. Ceci implique plus de points échangés mais une meilleure précision que les modèles d'ordres inférieurs.

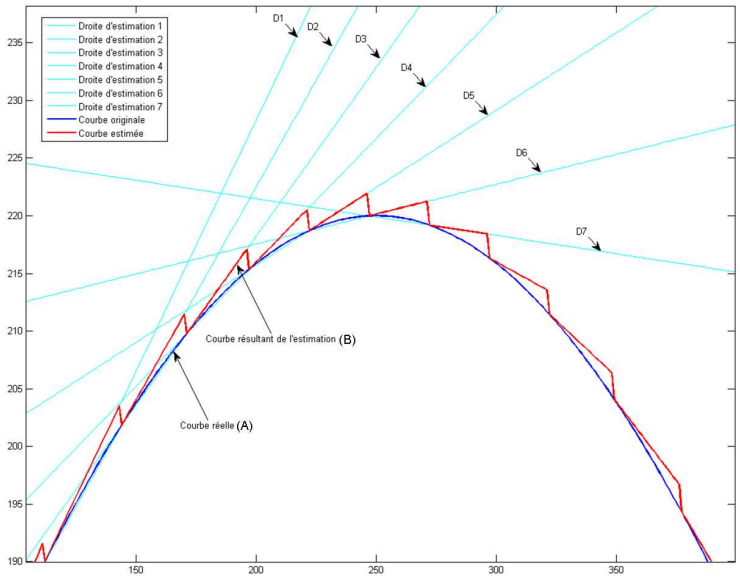

Figure 3.3 – Estimation d'une courbe par un polynôme d'ordre 1

En résumé, les estimateurs basés sur des polynômes d'ordre supérieur à 1 demandent d'échanger plus d'échantillons pour alimenter les calculs complexes afin de déterminer les coefficients du modèle. Un estimateur polynomial d'ordre 0 quant à lui, ne permet pas de suivre la tendance de la courbe entre l'envoi de deux points. Pour palier à ce problème, un polynôme d'ordre 1 permet par des calculs simples d'obtenir une meilleure estimation que le modèle précédent tout en rapportant la tendance de la courbe. Par ailleurs, ce modèle peut aboutir à un modèle simple d'ordre 0 suivant les données acquises. En conclusion, LiveNCM introduit la notion d'estimateur polynomial d'ordre 1, dans son schéma d'administration pour réduire les échantillons acquis. Les grandeurs physiques visées par cette technique sont celles qui évolueront lentement dans le temps comme le niveau de batterie ou la température, et qui toléreront une différence entre les mesures reportées et les mesures acquises.

Le principe de fonctionnement est alors très simple pour cet estimateur. Le noeud réalise les mesures et les compare avec le modèle d'estimation. Suivant le seuil fixé par l'utilisateur, le modèle décidera automatiquement si l'envoi de la mesure est pertinent. Si c'est le cas, le noeud met à jour son modèle d'estimation avant d'envoyer vers le sous-agent SNMP la dernière mesure acquise. Ensuite, ce sous-agent utilisera cette valeur pour mettre à jour son modèle. Au final, il considère toutes les valeurs qu'il reçoit comme des valeurs estimées, et ne connaît pas les valeurs réelles contrairement aux noeuds qui connaissent les deux valeurs. L'utilisateur aura à disposition une courbe estimée qui se rapproche des valeurs réelles avec une marge d'erreur. La définition de la tolérance est liée à la qualité de la grandeur que l'on souhaite observer et de la résolution du matériel de mesure. Par exemple, pour des applications environnementales, si

le matériel ne permet pas de réaliser une mesure au dixième de degrés, une marge de 1°C pour un relevé de température n'altérera que très peu le signal résultant. Néanmoins, il est toujours possible d'éliminer toute erreur associée à l'estimateur en fixant un seuil de 0. A ce moment, une seule mesure est envoyée tant qu'elle reste stable, et ensuite à la moindre variation tous les points sont envoyés. En complément de ceci, on peut se réserver la possibilité d'envoyer plus de points pour les périodes où la mesure reste stable. Ce mécanisme appelé 'synchronisation' permet de définir une période maximale entre deux mesures envoyées. L'objectif est de forcer le noeud à envoyer des données supplémentaires pour améliorer la précision des calculs d'estimation. Comme pour la tolérance, ce mécanisme peut être inhibé en mettant cette période à 0 ce qui pour le noeud revient à n'envoyer que les mesures pertinentes définies par le modèle d'estimation.

3.2.2 Le sous-agent SNMP : LiveNCM SubAgent

La piste de recherche retenue étant le protocole SNMP, nous avons choisi de développer un agent adapté aux besoins de gestion de données des différents RCSFs distants. Avant d'entrer dans les détails, il est important de rappeler les extensions possibles d'un agent SNMP classique. En effet, il existe trois moyens d'inclure de nouvelles données. La première consiste tout simplement à développer un agent complet, ainsi il sera nécessaire de reprendre la gestion des données génériques de la MIB-II [McCloghrie 91] par exemple. Ceci demande donc un travail important de réécriture mais aussi un travail d'adaptation pour chaque machine. Ce dernier point constitue donc un frein au développement de cette extension car elle sera inadaptée à nos besoins et difficile à porter sur différentes machines et OS. La deuxième extension consiste à développer un module associé à un agent déjà existant. Il se nomme "Proxy" et bénéficie de toute l'architecture de l'agent auquel il est rattaché. Malheureusement, une telle extension peut rendre l'agent parent instable au point de le rendre inutilisable en cas de fausse manoeuvre. Par ailleurs, le développement de cette extension s'effectue selon un squelette logiciel bien défini ce qui limite les nouvelles fonctionnalités. Pour finir, la troisième extension se rapproche de la précédente, mais elle s'appuie sur un module indépendant de l'agent parent. Ceci se traduit par un programme informatique qui étendra les données gérées et qui fonctionnera de manière autonome tant qu'un agent parent sera exécuté. En effet, avec cette architecture, l'agent parent reçoit les requêtes utilisateur avant de les diriger vers le sous-agent SNMP. Une telle extension offre une marge de développement plus élevée et une meilleure stabilité du système car elle est totalement indépendante du fonctionnement de l'agent parent. Le fonctionnement général de cette extension pourra être mieux adapté aux besoins avec un format de développement plus flexible. Un sous-agent de ce type reprend toute la partie correspondant aux traitements des requêtes SNMP tout en laissant à l'agent parent la gestion des autorisations ainsi que la communication avec l'utilisateur. Ici, une panne du sous-agent SNMP n'empêchera en aucun cas le bon fonctionnement de l'agent parent.

Au final, en reprenant les trois extensions possibles, la dernière, constituée d'un sous-agent autonome, semble être la meilleure solution de développement. LiveNCM s'appuiera sur un sous-agent SNMP déployé sur un serveur ou un ordinateur supportant déjà des fonctionnalités SNMP. Son rôle est de centraliser les données reçues d'un RCSF pour ensuite les mettre à

disposition de l'utilisateur. N'importe quel outil d'administration SNMP sera utilisé pour superviser les données du réseau. Que ce soit en local ou à distance, un utilisateur pourra gérer son RCSF de manière transparente. Le développement d'un agent SNMP s'accompagne de la définition d'une structure de données arborescente MIB dans laquelle on retrouvera l'ensemble des données supervisées sur le noeud.

Définition de la MIB employée par LiveNCM

Ce sous-agent SNMP aura pour rôle de centraliser les données de tous les noeuds de différents réseaux. Avec une telle vision, il est nécessaire de prévoir une architecture logicielle permettant la navigabilité nécessaire pour retrouver les liens parent/enfant entre les différents éléments (réseau-noeud-capteur). La dynamique des réseaux impose aussi de pouvoir modifier très facilement ces liens ainsi que la constitution de chacun de ces éléments. Malheureusement, il n'existe pas de structure de données dans SNMP permettant de réaliser ces deux fonctions. Les tables permettront d'avoir la modularité nécessaire pour faire évoluer la topologie, mais il n'y a aucun lien de parenté possible. Par ailleurs, pour garder les liens de parenté, il serait nécessaire d'avoir des tables de tables afin de pouvoir naviguer plus facilement. Or, le fonctionnement de SNMP ne permet pas cette structure de données car les primitives ne sont pas conçues pour y accéder. On trouve alors deux solutions pour répondre à ce problème. La première consiste à fixer dans chaque ligne d'une table, l'identifiant SNMP de l'élément parent. Pour chaque message que l'on souhaite envoyer vers les noeuds, il est nécessaire d'exécuter une requête SNMP supplémentaire pour accéder aux informations pour former et diffuser le message (adresse de la passerelle par exemple). L'architecture d'un agent SNMP ne permet pas de voir les données d'une branche à partir d'une autre. De plus, pour pouvoir exécuter une nouvelle requête SNMP et reprendre la précédente, un mécanisme de sauvegarde de contexte avec un automate devra être développé. Ce dernier point ne favorise donc pas le choix de cette solution car elle rendrait le sous-agent moins réactif et surtout le surchargerait avec un nombre de requêtes annexes important. La seconde solution consiste à maintenir un arbre secondaire qui surchargera la structure de données de SNMP par des informations sur les liens de parenté. Pour l'utilisateur, cette méthode est totalement transparente, et pour le sous-agent, il peut naviguer sur l'ensemble des noeuds d'un réseau pour accéder à certaines informations partagées. Lors d'une suppression, les structures de données 'enfant' d'un élément seront automatiquement effacées, c'est-à-dire que lorsque l'on supprimera un noeud, les capteurs seront automatiquement effacés de la base car ils n'ont plus d'existence physique dans le réseau. L'intérêt de cette solution est de n'impliquer aucune surcharge avec des requêtes annexes tout en améliorant la réactivité et les temps de réponses à une requête utilisateur. Cette solution a été modélisée par le diagramme Unified Modeling Language (UML) de la Figure 3.4. Sur ce diagramme, la partie de gauche représente la modélisation de la structure de données employée par le protocole SNMP pour gérer les données de type 'table'. La partie de droite quant à elle, propose une nouvelle structure 'row_container' permettant de maintenir les liens de parenté entre les différents éléments. L'idée est de réaliser un arbre reportant la topologie du réseau. Un réseau sera défini par un ensemble de fils traitant les informations des noeuds. Chacun d'eux aura des fils qui représenteront les différents capteurs embarqués. Les classes 'cell_container' et 'list_cell_container'

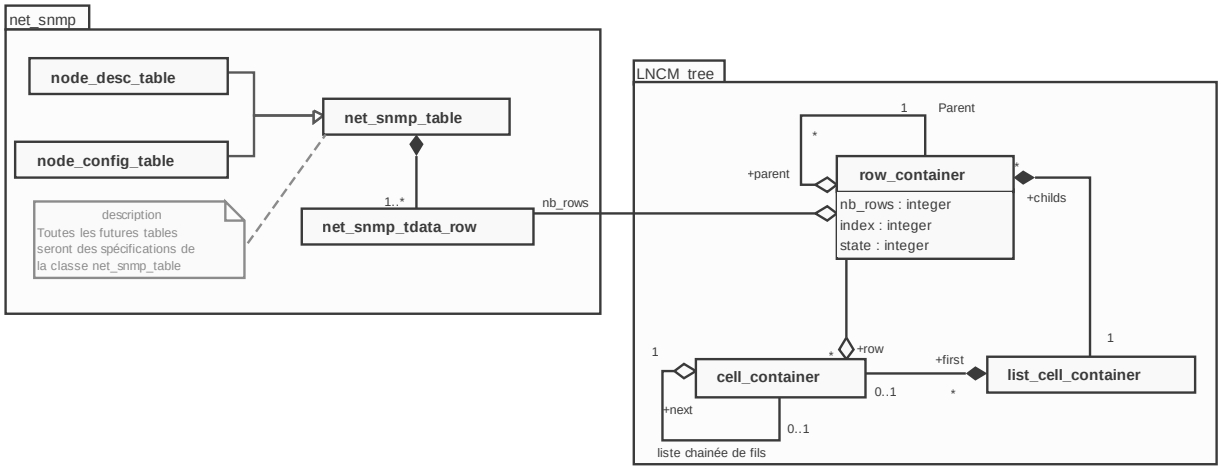

Figure 3.4 – Modélisation de la gestion de la parenté entre réseau, noeud et capteur

modélisent une liste chaînée permettant de regrouper les conteneurs fils d'un parent.

Cette nouvelle structure générique est reprise par tous les éléments supervisés. Il est donc possible de créer des groupes de RCSFs tout comme des groupes de noeuds. La classe générique 'row_container' agrège alors un ensemble de lignes du même type dans une table SNMP pour former un groupe. La navigation et l'accès aux informations partagées sont alors plus rapides et plus fluides avec une architecture hiérarchique de ce type. A chaque requête à destination d'un noeud, le sous-agent n'aura qu'à remonter dans l'arbre pour obtenir les informations nécessaires à la formation du message mais aussi à la diffusion du message. L'exemple typique est celui de la modification d'un paramètre sur un capteur. La requête portera sur la table décrivant les capteurs. Le sous-agent remontera alors d'un niveau pour connaître l'identifiant du noeud associé au capteur, et remontera encore d'un niveau pour connaître le moyen d'accéder au noeud.

Ensuite, pour le développement du sous-agent, les données de chaque élément (réseau-noeud-capteur) à superviser devront être définies au préalable. Pour cela, chacun d'eux disposera d'un ensemble de données regroupées sous forme de tables. Une première modélisation des données a donc été effectuée par un diagramme de classe UML présenté par la Figure 3.5.

On distingue 3 grands groupes de données concernant les RCSFs, les noeuds et les capteurs. Chacun d'eux est représenté par une ou plusieurs tables. Pour chaque groupe, une première définition des informations a été effectuée pour superviser les principaux paramètres d'un réseau, d'un noeud et d'un capteur.

Le groupe *réseau* dispose donc des informations concernant l'identification, le nom, la localisation, et la passerelle de communication. Sans ce dernier paramètre qui définit l'adresse de la passerelle, les échanges avec les noeuds sont impossibles car l'agent ne peut savoir comment joindre un noeud. On s'en servira donc pour valider l'existence d'un réseau via un message de contrôle par exemple. Un réseau sera donc assimilé à l'existence d'une passerelle de commu-

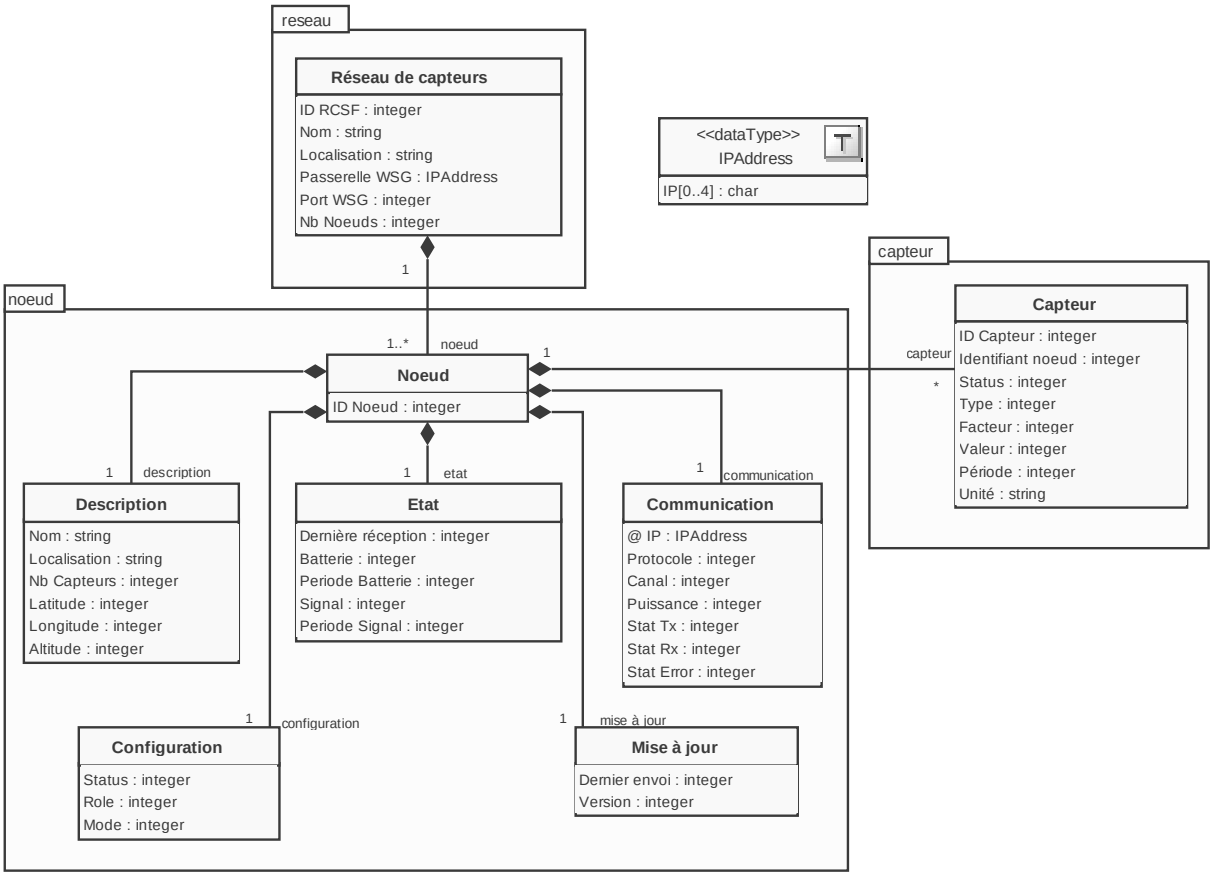

Figure 3.5 – Diagramme de classes UML des informations administrées par LiveNCM

nication. Ce paramètre très important est utilisé pour toutes les requêtes à transférer vers les noeuds. Il est donc nécessaire de partager cette information entre les différentes tables pour pouvoir envoyer des messages concernant aussi bien un noeud qu'un capteur.

Le groupe *noeud* regroupe plusieurs types d'informations concernant la description, la configuration, l'état, la communication, et les mises à jour. Le type et l'impact des paramètres définissent différentes tables :

- *Description* : Carte d'identité du noeud, contenant son nom, sa localisation, le nombre de capteurs et l'identifiant du réseau de rattachement.
- *Configuration* : Statut, mode de fonctionnement et rôle du noeud.
- *État* : Niveau des ressources du noeud (batterie, espace mémoire, signal reçu).
- *Mises à jour* : Gestion des informations de mise à jour du noeud.
- *Communication* : Paramètres et statistiques liés au médium de communication employé.

Le statut du noeud définit l'état courant de fonctionnement : en veille, actif, ou panne détectée par exemple. Le mode de fonctionnement permet de définir la politique de gestion des ressources avec un mode *'normal'*, un mode *'basse consommation'* où le noeud réduira sa fréquence de fonctionnement et éteindra certains périphériques, et un dernier mode *'sécurité'* qui annulera toutes les tâches actives sur le noeud. Dans ce dernier mode, le noeud ne fera que répondre aux requêtes d'administration ou sera complètement éteint en attente du renouvellement de sa source énergétique ou de la résolution de toute autre panne. Concernant la table communication, les paramètres que l'on y retrouve principalement sont l'adresse du noeud, la puissance de transmission ainsi que le canal d'émission. L'utilisateur aura donc la possibilité d'agir sur tous ces paramètres pour améliorer ou réduire les performances du noeud et ainsi contrôler l'énergie dédiée à la communication. Quelques statistiques concernant les échanges seront incluses dans ce groupe pour permettre de suivre les communications et détecter les noeuds mal placés d'un point de vue communication (taux de pertes des messages, ou d'erreurs de réception). Enfin, la dernière table 'update' sera principalement utilisée pour stocker la version du logiciel embarqué sur le noeud ainsi que des informations concernant la date de prise en compte. Elle sera maintenue par le noeud dès qu'il exécutera la première fois la nouvelle application, mais aussi par le logiciel de mise à jour à la fin de la séquence d'envoi et de contrôle.

Enfin, le groupe *capteur* s'occupe de gérer les paramètres de gestion du capteur, comme les coefficients de correction, la calibration ou le taux d'échantillonnage. La dernière valeur acquise par le capteur apparaît aussi dans un but informatif seulement. L'utilisateur devra développer tout traitement et tout historique des données car SNMP ne permet d'avoir l'image d'un système qu'à l'instant t. Les données seront donc écrasées à chaque mise à jour par le noeud. Les paramètres permettant de piloter les méthodes de réduction de données utilisées par LiveNCM font également partie de ce groupe.

Enfin, pour former un nouveau sous-agent, le protocole SNMP prévoit la définition des différents paramètres dans un fichier MIB en langage ASN.1. Généralement, pour décrire ces données, la MIB-II [McCloghrie 91] utilise seulement deux types simples de données : entier (INTEGER) et chaîne de caractères (OCTET STRING). D'autres types comme booléen ou réel ne sont pas gérés par le protocole SNMP ce qui rend plus complexe la gestion de valeurs

décimales. Néanmoins, des types plus évolués peuvent être employés pour définir une jauge (incrémentation ou décrémentation possible), un compteur (incrémentation uniquement possible), une adresse IP, ou des tics d'horloge (time-stamp de données). L'écriture de la MIB définit alors l'organisation hiérarchique des données ainsi que la description des différents objets. Des séquences d'objets définiront des tables qui regrouperont des informations dynamiques comme les interfaces réseau sur un ordinateur ou les capteurs sur un noeud. Les groupes définissent un ensemble de paramètres fixes et connus à l'avance. Chacun d'eux dispose d'un droit d'accès allant de la non visibilité jusqu'à la lecture et l'écriture, et d'un statut, *'obsolete'* pour la compatibilité entre les anciennes et nouvelles versions, *'optional'* pour les données facultatives ou obligatoire *'mandatory'* pour tout autre type de données. La structure des données est alors représentée par un fichier MIB décrivant la structure de la Figure 3.6. On retrouve alors les 3 groupes de données définis dans la modélisation UML de la Figure 3.5, réseau (wsn), noeuds (nodes) et capteurs (sensors).

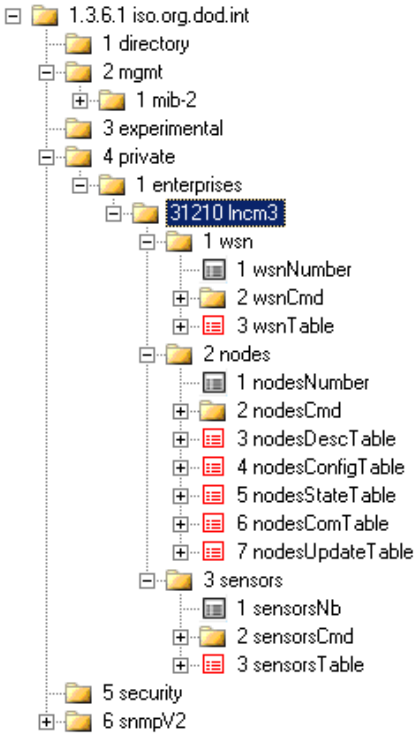

Figure 3.6 – Structure arborescente de la MIB employée par LiveNCM

Chaque paramètre dans la MIB est décrit par un type ASN.1 comprenant l'entier, le caractère ou la chaîne de caractères. Un entier représente 4 octets, un caractère 1 octet, et une chaîne de caractères 256 octets. Ainsi, la table représentant les données du RCSF 'wsn' utilise 536 octets, celles représentant les noeuds 'nodes' emploient 618 octets et celle réservée aux capteurs 'sensors' utilise 284 octets. Pour un réseau de dix noeuds comportant chacun un capteur, l'espace mémoire de stockage de ses données est de 9.6 ko.

Finalement, nous proposons ce découpage afin de définir des groupes de paramètres administrés suivant leurs impacts sur le noeud. Néanmoins, les paramètres inclus dans cette MIB ne sont pas exhaustifs. Il est toujours possible de les étendre facilement par de nouvelles branches et feuilles dans l'arbre. L'extension se traduira par une modification de la MIB en ajoutant les nouvelles informations formatées par le langage ASN.1 avec des identifiants uniques. Ensuite, quelques lignes supplémentaires dans le programme du sous-agent permettront de les gérer via le protocole SNMP. Sur les noeuds, l'impact d'un ajout passe par la définition d'une nouvelle commande ainsi que des traitements associés.

Persistance des données

Le protocole SNMP permet d'avoir l'état d'un système à l'instant t. De part sa conception, un agent sauvegarde uniquement cette image dans la mémoire qui lui est accordée pour son exécution. Or, cette zone de stockage ne persiste pas entre deux redémarrages de l'agent. Pour des équipements standards tels que les ordinateurs ou les imprimantes, ce détail n'a que peu d'incidence car l'agent est sur la machine et peut reconstituer les données de la MIB très rapidement. Dans notre cas, en choisissant un agent SNMP centralisé, les données sont perdues pour tous les noeuds du réseau à chaque arrêt du sous-agent ou de l'agent parent. Il ne sera plus capable de retrouver une passerelle de communication puisque son adresse est stockée dans cette MIB. Pour résoudre ce problème, le sous-agent SNMP développé intègre un module de sauvegarde de données. Il réalise des images périodiques de la MIB ainsi qu'au moment de son extinction. La sauvegarde est donc un point important de notre solution d'administration. C'est pourquoi, nous avons choisi d'utiliser des fichiers de sauvegarde intermédiaires. La syntaxe utilisée est celle du langage eXtensible Markup Language (XML) qui a été créée dans le but de faciliter le stockage des données et de permettre une utilisation par différentes applications. De plus, ce langage permet de définir ses propres balises afin de rendre le résultat compréhensible par n'importe qui. Les balises, ouvrante <ma balise> et fermante </ma balise>, définissent des données ou des groupes de données. Leurs définitions s'effectuent par l'intermédiaire d'un fichier Document Type Definition (DTD) explicitant un modèle de document XML. Une DTD indique les noms des éléments pouvant apparaître et leur contenu, c'est-à-dire les sous-éléments et les attributs. En dehors des attributs, le contenu est spécifié en indiquant le nom, l'ordre et le nombre d'occurrences autorisées des sous-éléments. L'ensemble constitue la définition des hiérarchies valides d'éléments et de texte. Une DTD a donc été créée pour définir les balises pour chaque branche et feuille de la MIB. Pour une meilleure lisibilité, ces balises porteront le même nom que l'élément qu'elles représentent. Ensuite, en s'appuyant sur cette DTD, le module de sauvegarde génère un fichier XML (Figure 3.7) reprenant l'ensemble des informations de la MIB.

La périodicité des sauvegardes est définie par l'utilisateur à l'aide du fichier de configuration du sous-agent. Cependant, pour ne perdre aucune information importante comme l'ajout d'un noeud ou d'un réseau, le module effectue des sauvegardes après certaines opérations critiques. Ainsi, si l'agent a une panne, il sera toujours possible de retrouver les derniers éléments ajoutés. Enfin, une dernière sauvegarde est effectuée à l'extinction du sous-agent par l'utilisateur ou dans le cas de certaines pannes. Ensuite au redémarrage, le fichier de sauvegarde est interprété par

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Chargement de la DTD décrivant les balises de cet exemple -->
<!DOCTYPE mib SYSTEM "/var/www/projet/xml/mib.dtd">
<mib>
    <agent>
        <version>0.2.0.4_DEV</version>
        <save_Date>1252334755</save_Date>
        <build>134633196</build>
    </agent>
    <list_WSNs>
        <!-- UN RCSF -->
        <WSN id="1">
            <WSNDesc>
                <WSNDescLocation>Cemagref</WSNDescLocation>
                <WSNDescName>Bureau</WSNDescName>
                <WSNDescGw>127.0.0.1</WSNDescGw>
                <WSNDescPort>4244</WSNDescPort>
                <WSNDescNbNodes>1</WSNDescNbNodes>
            </WSNDesc>
            <WSNNodes>
                <!-- UN NOEUD -->
                <node id="1">
                    <nodeDesc>
                        <nodeDescLocation>Fenetre</nodeDescLocation>
                        <nodeDescName>Node1</nodeDescName>
                        ...
                    </nodeDesc>
                    <nodeConfig>
                        ...
```

Figure 3.7 – Exemple de fichier XML de sauvegarde de l'architecture de la MIB

ce même module pour remplir la MIB. Il est possible de définir le nombre de sauvegardes intermédiaires à garder afin de pouvoir revenir sur certaines opérations qui peuvent poser des difficultés.

3.2.3 Le protocole de communication : LiveNCM communication Protocol

Dans la plupart des méthodes qu'elles soient d'administration, de mise à jour, ou de routage, il est indispensable de définir un protocole de communication, spécifique ou standardisé. L'inconvénient d'une vision orientée développement spécifique est de rendre difficile l'interopérabilité des solutions d'administration entre elles. Par contre, en utilisant un protocole standard, elles peuvent échanger entre elles sans traitement supplémentaire. Cependant, dans ce cas, la standardisation du format des messages oblige l'utilisateur à envoyer des informations avec un format inadapté ou trop gourmand. L'exemple de la définition d'une date est assez flagrant. Par définition, elle est diffusée par une chaîne de caractères de 13 caractères ("20090901T1300") alors que le noeud peut la traiter avec seulement 6 caractères (0x20 09 09 01 13 00). Dans le cas des RCSFs, l'envoi de données inutiles ou trop gourmandes constitue une perte énergétique. Par ailleurs, les travaux menés pour OpenGIS Sensor Model Language Encoding Standard (SensorML) se basent sur le langage XML pour échanger des informations sur les noeuds. Une telle vision demande alors de disposer sur les noeuds d'une application de traitements des fichiers reçus ce qui représente un espace mémoire réservé mais aussi une charge de calcul supplémentaire. De plus, avec un langage comme XML, de nombreuses informations

seront échangées sans rapport avec la donnée transmise ce qui constitue une perte énergétique. Néanmoins, cette formalisation des messages permet une réutilisation des messages par divers logiciels. Cet avantage peut être un inconvénient d'un point de vue sécurité car n'importe quelle personne peut intercepter les messages et en lire le contenu. Ce point rend donc ce langage difficilement utilisable sans avoir un minimum de sécurité dans le RCSF. Ensuite, les protocoles de communication existants comme 6LoWPAN autorisent une interopérabilité entre ces réseaux et les réseaux filaires. L'utilisateur utilise un noeud comme une machine sur un réseau distant. On autorise alors toute personne à interagir avec les noeuds. Cependant, le format des messages incorpore des paramètres qui ne sont pas utiles au noeud ni à la transmission (type de compression, d'entête, ou de données). Dans le RCSF, les messages n'ont besoin que d'un type de commande et d'un numéro de commande pour définir un ensemble d'actions sur le noeud et pour transmettre de nouvelles fonctionnalités. C'est pourquoi, les échanges de données se font ici par un protocole de communication, adapté aux contraintes de communication des RCSFs, nommé LiveNCM communication Protocol (LiveNCM-P), s'appuyant sur certaines caractéristiques des protocoles standards comme 6LoWPAN, ou Internet Protocol (IP). Pour cela, un nouveau format de message a été mis en place pour permettre une généricité des échanges et une extension facile tout en contrôlant l'intégrité des données transmises. L'objectif derrière ceci est de rendre le protocole de communication développé utilisable facilement pour tout type de messages, que ce soit pour le routage, la synchronisation, l'auto-configuration, l'administration ou tout simplement pour la collecte de données entre noeuds.

Format des messages

Il est important de rappeler que le plus gros consommateur d'énergie sur un noeud est le module de communication. En limitant les données à envoyer, on limite le temps de transfert et donc le temps de fonctionnement du médium. C'est pourquoi, un protocole de communication pour ces applications doit réduire et/ou compresser ses messages pour n'envoyer que les informations utiles. Il est ainsi nécessaire de voir les octets transmis comme une suite de bits qui peuvent coder une information binaire du noeud. Traditionnellement, les informations sont codées sur un octet même si une partie des bits n'est pas utilisée. Le travail de compression proposé par 6LoWPAN permet d'éliminer les bits inutiles en les optimisant pour transmettre une information. En outre, les informations des entêtes de message sont également codées. Cette technique de compression a donc été une base de travail pour définir les messages échangés par LiveNCM (Figure 3.8).

L'entête et la fin du message sont codées sur ce principe, c'est-à-dire que les données transmises ne sont plus lues comme des octets mais comme une suite de bits donnant des informations sur le message. On retrouve alors des informations standard telles que les adresses de la source et de la destination codées sur 2*32 bits. Ce codage revient alors à utiliser un adressage de type IPv4 pour joindre les noeuds. Ensuite les données concernant le routage du message, d'un point de vue de l'administration, sont codées avec quelques bits (nombre de sauts sur 4 bits, priorité sur 2 bits). L'information 'nombre de sauts' se rapproche du Time To Live (TTL) du protocole IP ce qui permet de limiter la portée d'un message dans un réseau. A chaque relais, le noeud le décrémente et lorsqu'on reçoit un message avec un nombre de sauts nul, soit on le traite, soit on

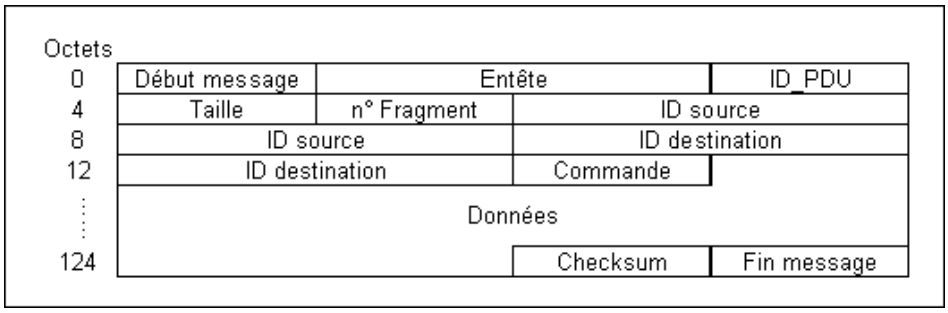

Figure 3.8 – Format des messages échangés

le supprime. Il est donc nécessaire de connaître à l'avance le parcours du message pour éviter qu'il ne soit bloqué avant d'arriver au(x) destinataire(s). La connaissance du chemin passe par une solution de routage efficace permettant de disposer du nombre de sauts à effectuer pour aller d'un point A à un point B. Cette partie complexe n'entre cependant pas dans le domaine de l'administration et sera confiée au protocole de routage utilisé dans le réseau. Par ailleurs, la priorité d'un message constitue un point délicat de l'administration. La définition de la priorité d'un message pour l'administration est basée sur la rapidité souhaitée pour le traitement de la requête, sur l'importance des données transportées et enfin sur la qualité de service désirée pour la transmission. On distingue donc 4 niveaux de priorité :

- *Basse* : Les messages sont exécutés après les autres requêtes plus prioritaires. Les messages de demande d'information comme les statistiques auront cette priorité.
- *Normale* : La requête est traitée à la réception. Les messages courants sont envoyés avec cette priorité.
- *Supérieure à la normale* : Les données sont relativement sensibles, et doivent être traitées le plus rapidement possible. Des données sensibles ou des traitements importants sont envoyés avec une telle priorité pour que le noeud réagisse rapidement à la demande.
- *Haute* : Le traitement des informations reçues doit se faire immédiatement. Pour cela, soit on termine la requête en cours, soit on la stoppe pour la reprendre plus tard. Les alertes (intrusion, batterie critique, panne) et/ou les données directement liées à la vie du noeud ont une priorité haute, car dans bien des cas, on évite de perdre le noeud en réagissant rapidement en modifiant son mode de fonctionnement par exemple.

Par ailleurs, avec ces niveaux de priorité, une méthode additionnelle de contrôle de la QoS pourra modifier les différents paramètres d'envois pour améliorer les transmissions et garantir les échanges entre deux noeuds. Les messages seront insérés dans la pile de réception de la fin (priorité basse) vers le début (priorité haute) (Figure 3.9). Le traitement des requêtes se fera dans le sens inverse tout en veillant à ce que des messages ne restent pas indéfiniment dans la pile.

Dans un souci de préservation de la mémoire embarquée sur les noeuds, cette file de message sera assez courte. Les noeuds seront configurés pour accepter quelques messages tandis que les suivants seront perdus. Un ordonnanceur des messages est nécessaire pour gérer cette file. Son

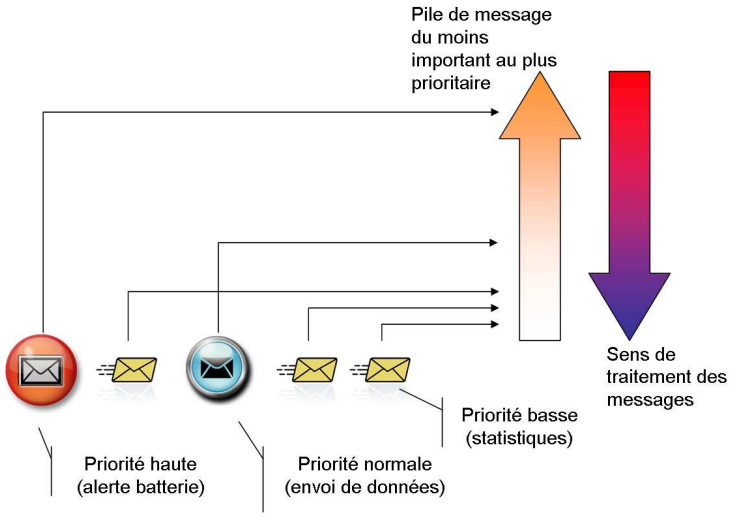

Figure 3.9 – Réception et traitement des messages en fonction de leur priorité

principe peut être simple en ne cherchant que les messages de plus forte priorité dans la file, ou plus complexe en gérant un ordre d'arrivée. Dans le premier cas, un message de faible priorité peut rester longtemps dans la file si le noeud reçoit plusieurs messages plus prioritaires, tandis que dans le second cas, les messages seront traités au maximum après un temps défini. LiveNCM intègre un ordonnanceur simple pour gérer les files d'attente de message sur les noeuds mais aussi sur la passerelle. L'évolution vers la solution complexe consiste à revoir la méthode de sélection de messages dans la file, le reste ne changeant pas.

En résumé, les premiers octets de chaque message contiendront un caractère de début de trame, des informations concernant le nombre de sauts et la priorité du message. Une évaluation simple du niveau de batterie sur 2 bits est aussi contenue dans l'entête pour définir 4 niveaux d'énergie (normal, moyen, bas, critique) basés sur la mesure de la tension d'alimentation du noeud. De plus, un octet est réservé pour permettre la fragmentation des messages. En effet, dans les réseaux de type 6LoWPAN, la taille des paquets peut être limitée à 127 octets sans fragmentation de la part du médium de communication [Montenegro 07]. Il est donc judicieux de prévoir un mécanisme pour diffuser des données de taille plus conséquente. Cet espace permet alors de diffuser des messages de 255 fragments de 110 octets (17 octets d'entête et de pied de message). En théorie, il est possible de diffuser des messages jusqu'à 28 ko. Quand ce paramètre vaudra 0, cela signifiera que le message est diffusé en une seule fois. On trouvera ensuite le type du message codé sur 4 bits ce qui permet d'avoir 16 types de requêtes sur les noeuds. Les premiers types définis regroupent les commandes associées à l'administration (ADM), aux données (DATA), au système de mise à jour (FILE), ou alors au protocole de communication. Les commandes de type administration accèdent ou modifient un paramètre associé au noeud, à sa configuration ou à son capteur. La catégorie 'données' permet d'obtenir les valeurs des différents capteurs disponibles sur le noeud ou de modifier leurs fréquences de mesure. De plus, le protocole de routage disposera lui aussi de ses propres commandes. Enfin,

le système de mise à jour détiendra un type spécifique de message pour échanger les différentes versions d'applications ainsi que des traitements de contrôle et de vérification. D'autres types de commandes seront implémentés afin de synchroniser les noeuds, d'alerter d'une anomalie, ou bien pour fournir à l'utilisateur un bouquet de commandes. Le numéro de commande est transmis sur un octet ce qui permet d'en définir au maximum 256 par type de message. En théorie, un noeud peut donc disposer de 4096 requêtes différentes (16 types de 256 commandes). En pratique, on choisira de réserver les 10 premières commandes de chaque type pour des traitements communs tels que la gestion d'acquittement, d'erreur, de sauvegarde ou d'extinction. Ensuite, chaque commande traitera un nombre d'arguments variables de types différents. Le contrôle des arguments sera effectué au moment du traitement de la commande et non pas à la vérification du message. Cette vérification consiste à valider l'intégrité du message lors de la transmission.

Le format de paquet défini par LiveNCM-P est constitué de toutes les informations nécessaires pour avoir un routage simple des messages dans le réseau. De cette façon, ce protocole de communication est complètement indépendant de toutes méthodes existantes. Cependant, comme indiqué dans le chapitre 2, le routage constitue un moyen de simplifier les échanges et d'améliorer l'accessibilité des noeuds. Ainsi, une reformulation des messages sera nécessaire pour coller à un protocole de routage plus complexe. Cette opération s'appuiera sur les informations déjà présentes comme les adresses ou le nombre de sauts, pour former les nouveaux paquets à diffuser. Dans le cadre de l'utilisation de 6LoWPAN comme pile de protocoles, l'adressage et le Cyclic Redundancy Check (CRC) des paquets LiveNCM-P disparaîtront pour être inclus dans les protocoles de plus bas niveau. En définitive, la reformulation consistera à enlever l'adressage des paquets LiveNCM-P pour les inclure dans les protocoles de plus bas niveau. Par ailleurs, les paquets sont définis à la taille maximale acceptable. L'ajout de protocole de plus bas niveau avec leurs entêtes aura pour effet de réduire le champ de données disponibles. Ainsi, avec 6LoWPAN, l'entête IPv6 réduira le champ de données de 2 octets. Ensuite, l'adressage utilisé réduira le champ de données de la taille des adresses employées (jusqu'à 8 octets par adresse, soit 16 octets pour la source et la destination). La taille maximale des paquets LiveNCM-P passe alors, en utilisant 6LoWPAN et User Datagramme Protocol (UDP) (48 octets d'entêtes), à 102 octets. L'entête LiveNCM-P passe quant à lui de 17 octets à 8 octets soit l'adressage et le CRC en moins.

Détection de nouveaux messages

Pour détecter les messages entrants, le protocole LiveNCM-P propose d'utiliser les 4 premiers octets reçus pour reconstruire le caractère de début de paquet. La Figure 3.10 présente la répartition des bits utiles et leur correspondance.

Ainsi les deux bits de poids fort des quatre premiers caractères reçus doivent pouvoir reconstituer le caractère de début de trame. Cette technique a pour inconvénient de fixer 2 bits sur ces caractères mais par contre la détection de message reçu est bien meilleure. Une suite de 4 octets sans contrainte définit un ensemble de 2^32 suites possibles. Ici, avec les contraintes établies, à savoir le premier octet est fixé puis les 2 bits de poids fort sont définis pour les 3 suivants, le nombre de suites possibles est réduit à 0.006% de l'ensemble des combinaisons

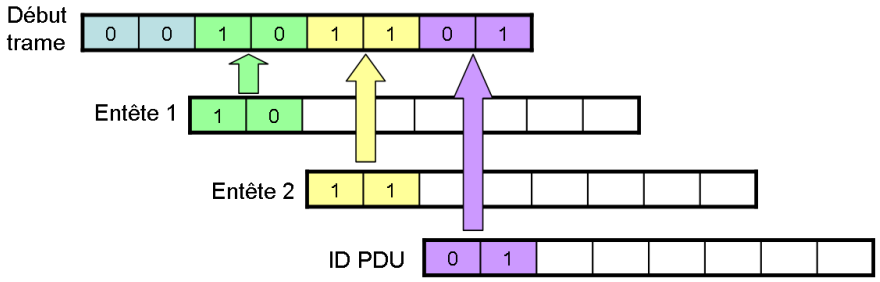

Figure 3.10 – Détection d'un nouveau message

possibles. Ainsi, la probabilité de détecter un mauvais message est déjà très faible. Cependant, on complète les messages par un caractère final tout aussi spécifique que le caractère de début de trame. La fin du message est détectée avec la taille donnée par le message, mais aussi en vérifiant son caractère de fin de trame, qui fixe les 2 bits poids forts (b7-b6) et faibles (b1-b0) à 1. Il existe alors 16 possibilités pour terminer un message. En reprenant les calculs de probabilité précédents, la définition de l'entête et du caractère de fin représente 0.0004% des messages possibles ce qui permet d'avoir une détection de messages très sûre. Ce principe très simple permet donc de détecter les messages reçus et de valider leur format (entête, taille) assez facilement avec un traitement peu coûteux en ressources mémoire ou processeur. La vérification de leur intégrité demandera, par contre, un minimum de ressources pour calculer un checksum CRC ou toute autre parité. La validation se fera à un niveau inférieur par l'intermédiaire du protocole de routage par exemple. Le message arrivant à l'application est alors considéré comme intègre d'un point de vue transmission. Les vérifications d'usage auront lieu au moment du traitement du message.

Par la suite, une fois le message reçu et validé, une première passe permet de distinguer l'auteur et le destinataire. Si le message est pour le noeud récepteur, alors une deuxième passe est effectuée pour traiter la commande et les arguments de la trame. Dans le cas contraire, le message est retransmis en broadcast vers ses voisins. La première étape est introduite afin d'être indépendante de tout protocole de routage. Néanmoins, pour des réseaux en utilisant un, les messages seront véhiculés du point A vers le point B sans nécessité de traitement de l'application. En résumé, le protocole LiveNCM-P développé permet de réaliser un routage très simple par inondation du réseau. A chaque noeud, une vérification d'intégrité du message est réalisée avant de le faire suivre. Ainsi, on limite la diffusion de messages erronés au sein du réseau. L'inondation du réseau est une technique simple mais qui multiplie les messages échangés. Le noeud maître de chaque cluster vérifie si le destinataire est l'un de ses noeuds esclaves, et dans ce cas, il lui transfère sinon il envoie aux clusters voisins. En revanche, pour spécifier aux noeuds de traiter un message spécifique, LiveNCM-P introduit une adresse de broadcast qui sera interprétée par les noeuds comme leur propre adresse afin de réaliser une opération de maintenance générale. L'exemple typique peut être celui de la synchronisation des noeuds. Cette action devra être traitée par tous les éléments du réseau sans que le noeud maître ou le superviseur n'ait à formuler 'N' messages pour 'N' noeuds. Pour finir, le protocole

LiveNCM-P est utilisé pour transmettre des données du sous-agent SNMP vers les noeuds.

3.2.4 La passerelle de communication : WSG

La liaison entre les noeuds et l'agent passe par l'ajout au réseau d'une interface supplémentaire que nous avons choisi de nommer passerelle Wireless Sensor Gateway (WSG). Le rôle de cet élément clé est de rendre accessible le RCSF au sous-agent SNMP. Le transfert des messages de l'un vers l'autre est son objectif premier en s'appuyant sur le protocole de communication LiveNCM-P développé. Pour illustrer cette interface, la Figure 3.11 représente ses entrées et sorties de données ainsi que la gestion du médium de communication.

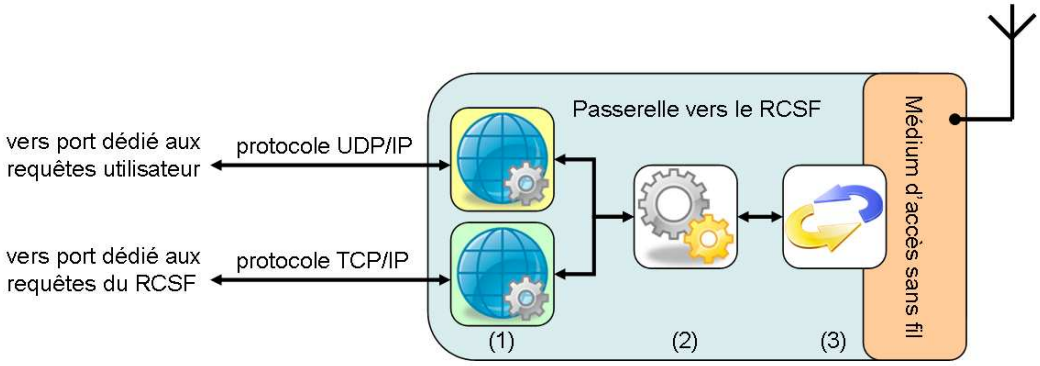

Figure 3.11 – Schématisation de la passerelle développée WSG : (1) traitements des messages, (2) vérification et pré traitements, (3) interface de communication avec le médium d'accès sans fil

Les échanges avec la passerelle sont effectués à travers des protocoles réseau standards (TCP, UDP) afin de la rendre accessible depuis n'importe quel point du réseau. Il est donc tout à fait possible de déployer le sous-agent SNMP et la passerelle en deux lieux géographiquement distants dialoguant par Internet ou tout autre réseau inter-connecté. L'adoption de cette communication a été aussi motivée par leurs fonctionnements et leurs capacités à garantir la diffusion. TCP implémente un mécanisme de synchronisation robuste qui permet de confirmer la bonne réception du message par le destinataire. De plus, TCP fonctionne en mode connecté et synchrone, c'est-à-dire qu'il y a forcément un émetteur et un récepteur contrairement à UDP qui lui est asynchrone, c'est-à-dire qu'il n'y a pas d'acquittement des messages par le récepteur et donc pas de connexion émetteur/récepteur. Les messages reçus sur ces interfaces peuvent ensuite faire l'objet de prétraitements ou alors être envoyés directement vers l'interface gérant le médium d'accès sans fil. Le chemin inverse est aussi valide, les remontées d'informations peuvent être agrégées afin de limiter les envois vers le sous-agent par exemple. Chaque message transitant est donc vérifié pour valider son intégrité avant de l'envoyer vers les noeuds. Cette vérification permet d'éliminer les messages mal formatés ou contenant des erreurs avant d'accéder aux noeuds et donc de consommer de l'énergie inutilement.

De part sa position dans le schéma d'administration, la passerelle WSG contiendra des traitements périodiques pour lesquels le sous-agent SNMP n'est pas nécessaire. Elle aura la tâche, par exemple, de synchroniser les noeuds ou d'appliquer la méthode d'auto-configuration choisie. Une grande variété d'applications peut être embarquée pour effectuer des actions réflexes à certains messages ou pour appliquer certaines tâches de gestion du réseau. Par exemple, lorsqu'il y a une alerte sur un niveau de batterie, la passerelle peut alors agir immédiatement sur le noeud pour passer dans un mode de fonctionnement plus économe. Le sous-agent SNMP définira une politique de gestion énergétique qui sera déployée et appliquée par la passerelle.

Auto-configuration de l'adressage et inscription d'un noeud

Avec un nombre important de noeuds, de quelques dizaines à plusieurs milliers, l'accès à chacun d'eux passe nécessairement par un adressage et une inscription automatique auprès du sous-agent SNMP. Pour cela, nous allons voir, dans un premier temps, qu'elle a été la réponse apportée pour les réseaux filaires. La solution choisie nommée Dynamic Host Configuration Protocol (DHCP) permet de fournir et configurer l'adressage d'une machine, de manière totalement transparente, pour l'utilisateur. Le principe est assez simple, la machine venant de se connecter diffuse sur le réseau une demande de configuration. Lorsqu'un serveur la reçoit, il cherche une adresse libre dans une plage définie par l'administrateur, puis l'envoie vers la machine. Celle-ci dispose ainsi de toutes les informations nécessaires pour accéder aux différents services proposés sur le réseau (messagerie, internet, serveur de fichiers, etc.). Le principe est donc intéressant pour adresser automatiquement les différents noeuds d'un RCSF. LiveNCM reprend cette technique pour l'adapter aux contraintes de communication mais aussi énergétiques. Le mécanisme mis en place est alors très simple. Le noeud nouvellement connecté broadcast un message HELLO dans lequel est intégré un identifiant unique associé matériellement au noeud. Toutes les passerelles recevant ce message y répondent par un acquittement ACK contenant leur adresse. Celle qui aura été la plus rapide, c'est-à-dire la plus proche du noeud ou la plus puissante, sera choisie par le noeud comme parent direct. Il sera associé à celle-ci et ne pourra recevoir plus aucun message des autres passerelles du réseau. Ensuite, le noeud demande son inscription auprès du sous-agent SNMP. La réponse se traduit sous la forme d'un message fixant l'adresse du noeud en fonction de l'identifiant réservé dans la MIB. Le noeud dispose donc des éléments suffisants pour permettre l'accès à ses paramètres par l'utilisateur. La Figure 3.12 présente la chronologie des échanges pour configurer l'adresse du noeud via son inscription dans la MIB.

Les messages HELLO envoyés par le noeud ne produisent pas d'état bloquant sur les passerelles, c'est-à-dire que celles-ci ne produisent qu'une action réflexe à ce type de message et n'attendent rien par la suite. L'ensemble des messages utilisés dans cette phase est du type administration. A chaque message correspond une commande définie qui pourra être complétée par un message d'acquittement. A la suite de l'inscription, les messages suivants envoyés automatiquement définiront les différents paramètres de communication et de configuration du noeud dans la MIB.

Enfin, quand un noeud détecte une perte de connexion avec sa passerelle, c'est-à-dire quand il ne reçoit plus de synchronisation ou ne reçoit plus de réponse à ces messages, il pourra relancer la

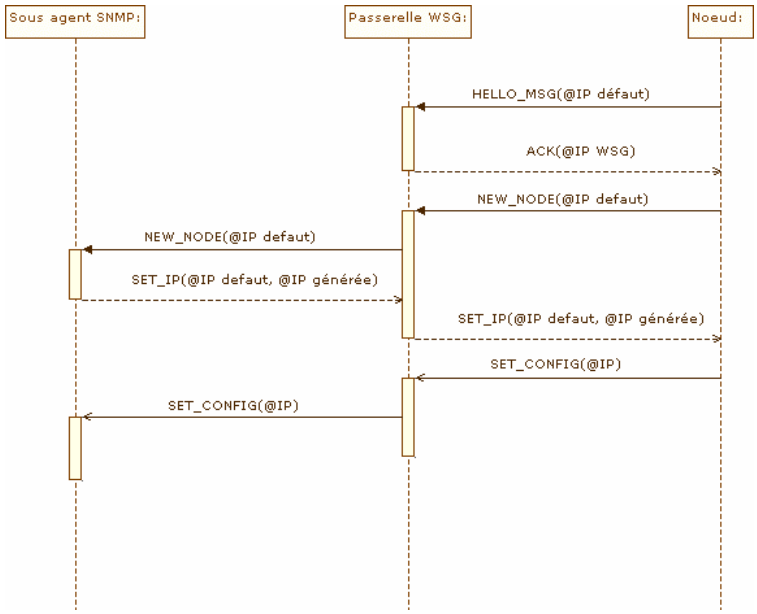

Figure 3.12 – Diagramme de séquence de la configuration automatique de l'adressage des noeuds

procédure précédente afin de se connecter à une autre passerelle. Le protocole de communication développé prévoit le changement de passerelle à travers une commande de réinscription du noeud. Dès qu'il se connectera à une nouvelle passerelle, il demandera à l'agent SNMP de migrer ses informations vers son nouveau réseau. En pratique, il suffit juste de changer l'identifiant du réseau associé au noeud dans la table 'description' et le reste des informations dans la MIB suivra. Pour traiter la problématique des noeuds mobiles, une méthode de synchronisation efficace permettra une meilleure évolution de la topologie du réseau.

Synchronisation des noeuds

La synchronisation des noeuds est une étape très importante dans la vie du réseau. Sans ceci, la mise en place de mesures périodiques et datées est impossible et les échanges entre les noeuds et le superviseur deviendront compliqués surtout si le noeud se réveille à période fixe par exemple. Les fenêtres d'envoi n'étant plus alignées, le noeud peut s'éteindre alors que le superviseur a des requêtes à lui envoyer. Le point commun étant la passerelle, ce sera donc à elle de fournir une date universelle pour tout le réseau. Sur le principe, la solution à apporter est assez simple. Un message est envoyé à tous les noeuds avec comme paramètres les informations de synchronisation. A la réception de ce type de message, le noeud appliquera immédiatement les modifications sur son horloge interne afin d'être en phase avec la passerelle. Dans la pratique, les nombreuses perturbations que peuvent subir les noeuds entraînent nécessairement la perte de message dans le réseau. Ainsi, certains noeuds ne recevront pas de mise à jour de leur horloge. En prenant en compte ce fait, la synchronisation des noeuds devient alors plus complexe. En effet, chaque noeud devra confirmer la bonne réception du message s'il le reçoit. Dans le cas

contraire, le noeud pourra lui aussi forcer sa synchronisation. De cette manière, les noeuds ont de meilleures probabilités d'être à l'heure vis-à-vis de la passerelle. Les RCSFs demandent donc de disposer de méthodes et mécanismes permettant de garder les noeuds ordonnés entre eux afin d'optimiser les temps de fonctionnement. Pour certains échanges, collecte de données par exemple, le temps de fonctionnement d'un noeud ne dépasse pas la seconde. Avec des fenêtres d'échanges synchrones, on maximise les chances d'optimiser les transferts de données et donc les temps de fonctionnement des noeuds.

Politique énergétique des noeuds

Les noeuds des RCSFs sont connus pour être des systèmes embarquées à fortes contraintes matérielles, logicielles et environnementales. Les déplacements pour traiter les pannes ne sont pas des plus faciles et quand il est possible d'accéder au noeud, il peut être impossible d'ajuster certains paramètres d'alimentation ou de changer la source énergétique. Par ailleurs, le cahier des charges d'un RCSF stipule généralement une durée minimale d'autonomie des noeuds. Pour répondre à ces points, une politique de gestion énergétique est la bienvenue sur les noeuds pour adapter leur comportement aux besoins. La solution LiveNCM intègre donc une méthode adaptable pour agir sur les différents modules d'un noeud à travers différents modes de fonctionnement. Chaque mode agira sur des paramètres plus ou moins importants du noeud pour limiter la consommation énergétique des différents éléments. L'ajustement de la fréquence de fonctionnement du processeur est une piste intéressante afin de jouer sur les consommations des différents périphériques embarqués dans le processeur. La gestion de réveils et de l'échantillonnage des données sont des techniques qui permettront de réduire les temps de fonctionnement et le trafic sur le médium d'accès sans fil. La compression de données peut alors être utilisée pour réduire les données échangées sur le réseau pour un temps. Des estimateurs pourront être aussi appliqués et adaptés pour n'envoyer que les données utiles en fonction d'une marge d'erreur. Le programme de gestion d'énergie jouera alors sur les paramètres de l'estimateur afin de contrôler les mesures envoyées. Certaines plateformes matérielles embarquent un module dédié à la gestion énergétique permettant de réduire la consommation en jouant sur les alimentations des différents éléments. La possibilité d'éteindre le noeud complètement est alors présente ce qui demande de prévoir des périodes de réveil. Pour pouvoir appliquer la politique énergétique mais aussi pour pouvoir dialoguer avec les noeuds, une synchronisation devient alors nécessaire pour rattraper toute dérive temporelle.

3.2.5 Gestion des données d'administration

Les contraintes d'administration sur les RCSFs imposent un faible stockage des données sur le noeud. La réponse est donc de fournir des méthodes pour interroger les différents éléments d'un noeud afin de répondre aux requêtes des utilisateurs. Cependant, un stockage de certains paramètres reste nécessaire afin de réduire le nombre d'accès à un élément, par exemple pour obtenir l'adresse du noeud, et réduire le temps de réponse du noeud. Ainsi, une structure de données regroupe les informations utilisées fréquemment comme l'adresse du noeud, les statistiques ou les paramètres des estimateurs. Le noeud dispose alors d'une image simplifiée

de sa configuration et de son état courant. L'état courant ainsi que le mode de fonctionnement feront partie de cet ensemble tout comme les informations de routage ou le rôle du noeud dans le RCSF. La contrainte de mémoire liée à l'administration impose donc de restreindre ces paramètres au maximum. La structure de données ainsi créée devra persister durant toute la vie du noeud, c'est pourquoi des méthodes de sauvegarde et de restauration doivent être mises en oeuvre. La sauvegarde s'effectue sur une mémoire interne fixe (Flash) et dispose d'un contrôle d'intégrité des données. Afin de limiter les algorithmes de contrôle, il peut se baser sur le même contrôle d'erreur que celui intégré aux messages. L'intérêt d'ajouter ce contrôle est de vérifier la validité de la sauvegarde et de sa restauration. Une erreur détectée à ce niveau entraînera la vérification des différents paramètres par le noeud en interrogeant ses éléments ou en les demandant au sous-agent SNMP. Par ailleurs, la restauration de cet ensemble d'informations n'a pas pour objectif de réinitialiser les éléments du noeud. L'image du noeud sauvegardée à l'instant t sera toujours valide à l'instant t+1, c'est-à-dire au prochain réveil.

3.2.6 Implémentation et déploiement de la passerelle WSG

La passerelle de communication pourra selon son utilisation prendra différentes formes logicielles et matérielles. Pour les illustrer, la Figure 3.13 représente les implantations possibles de la passerelle de communication.

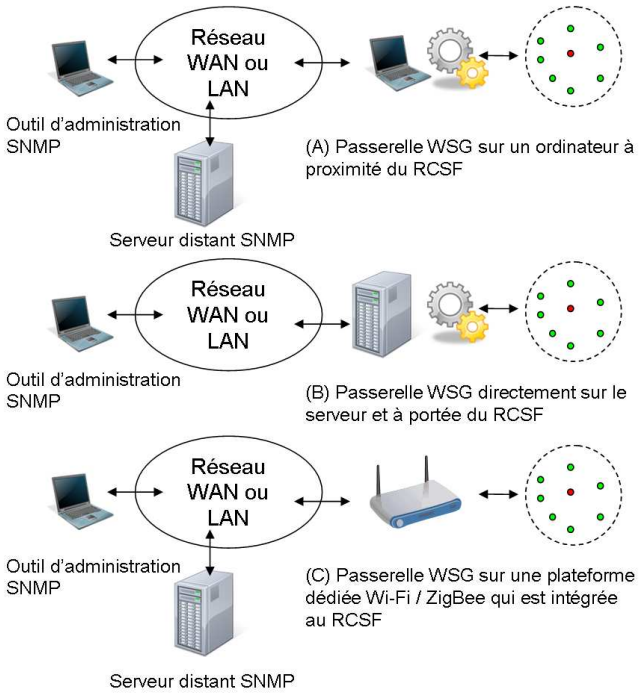

Figure 3.13 – Exemple de déploiement de la passerelle de communication

Dans les cas (A) et (B), la passerelle de communication est matérialisée par un logiciel se

chargeant de transférer les données d'un médium à un autre. Le premier exemple utilise un ordinateur pour l'héberger à proximité du RCSF. Le second exemple embarque l'agent SNMP et la passerelle sur un même ordinateur. Enfin dans le troisième cas (C), la passerelle est directement implémentée sur une station qui fera partie intégrante du RCSF. En choisissant cette option, elle devra gérer toutes les interfaces de communication. Dans cette configuration, elle devra disposer d'un moyen efficace de joindre un réseau fixe. Si ce n'est pas le cas, il reste la solution d'employer une solution Global System for Mobile communication (GSM) en passant par les réseaux de téléphonie mobile existants. Dans cette déclinaison, on se rapproche alors du cas (B) de l'exemple, car le module GSM sera associé directement au serveur de données. La différence principale entre les cas ((A), (B)) et (C) porte sur les ressources disponibles pour exécuter la tâche de transfert des messages. Dans les deux premiers cas, elles sont illimitées (à l'échelle d'un noeud) et le programme n'aura qu'à gérer l'interface de communication sans fil avec le réseau. Les temps de traitement d'une requête pourront être négligeables car ils seront absorbés par les buffers internes à l'interface de réseau de l'hôte, ou par un surdimensionnement des buffers circulaires. Dans le cas (C), les ressources utilisables sont limitées ce qui obligera à optimiser les codes embarqués et donc à limiter la taille des buffers de réception. De plus, l'application aura à traiter rapidement les informations venant de 2 à 3 interfaces différentes ce qui augmentera les temps de traitement d'une requête. Dans ces trois cas, le programme employé pour définir la passerelle devra comporter un maximum de parties communes à tout système d'exploitation et tout type de noeud pour favoriser la portabilité et ainsi améliorer les temps de développement pour passer d'une architecture à une autre. Quelques paramètres seront à adapter selon la plateforme matérielle employée pour supporter la passerelle de communication.

3.3 Module de mise à jour à distance

Les RCSFs sont des systèmes qui doivent pouvoir évoluer au fil du temps que ce soit en terme de nouvelles fonctionnalités ou en techniques de traitement du signal acquis. Pour cela, la réponse courante est d'intégrer un module de mise à jour à distance du logiciel embarqué. Son objectif est de prendre en compte de nouvelles applications répondant à une nouvelle fonctionnalité. Cependant, la diffusion de l'application n'est pas aussi facile. La synchronisation des noeuds entre eux est nécessaire afin d'avoir des échanges synchrones entre le superviseur et le noeud concerné, et de migrer tous les noeuds à la nouvelle version du logiciel en même temps. Par ailleurs, l'utilisateur doit pouvoir garantir les données échangées avant que le noeud puisse prendre en compte la mise à jour. On traduira ceci par un mécanisme de contrôle des messages reçus et un mécanisme de contrôle de l'intégrité du programme complet. De plus, les volumes de données échangées pouvant être importants, il est intéressant d'intégrer un algorithme de compression. Dans cette section, nous allons voir comment LiveNCM répond à cette problématique qu'est la mise à jour à distance des noeuds. Elle s'appuiera sur une technique reconnue de contrôle d'intégrité de fichier, l'algorithme Message Digest 5 (MD5) [Rivest 92], ainsi que sur des algorithmes de compression du type Huffman [Huffman 52] ou LZW [Welch 84].

3.3.1 Principe de fonctionnement

L'objectif de ce module est bien entendu d'envoyer et de mettre à jour tout ou partie d'une application. Les premières opérations seront la création des messages de mise à jour en optimisant leur nombre. Les contraintes imposées par les RCSFs impliquent d'effectuer un découpage de l'application en blocs de taille fixe (110 octets en utilisant LiveNCM-P). Cette étape sera effectuée par un logiciel externe qui s'appuiera sur la passerelle de communication pour diffuser le programme. Une communication de type Transmission Control Protocol (TCP) sera privilégiée afin de garantir la transmission et la bonne réception des différents messages par la passerelle. Pour cela, la passerelle de communication WSG doit être adaptée pour permettre au logiciel de mise à jour d'accéder directement au noeud. Ce dernier devra optimiser le nombre de messages nécessaires à l'envoi de la mise à jour tout en garantissant la réception intégrale des données. La mise à jour s'effectuera donc en plusieurs étapes décrites par l'organigramme de la Figure 3.14. La partie de gauche décrit la suite des actions menées par le logiciel de mise à jour. La première grande étape consiste à compresser le fichier binaire et à le fragmenter pour l'envoyer en une suite de messages. Chaque message reçu correctement par le noeud sera acquitté afin de débloquer le message suivant. En cas d'erreur ou si la réponse est hors délai, le logiciel retransmettra ce même bloc. La perte de paquets avec ce système se traduit par la non réception du message d'acquittement. Dans ce cas, le logiciel de mise à jour retransmettra le dernier message jusqu'à réception d'un acquittement ou à défaut jusqu'au nombre de retransmissions défini. De même, la panne d'un noeud revient au cas précédent de pertes de messages, et dans ce cas la mise à jour sera reportée au prochain réveil du noeud ou à la prochaine étape de mise à jour. Enfin les messages d'acquittement servent à libérer le bloc de données suivants. Un noeud ayant correctement reçu un bloc n'enverra qu'un seul message d'acquittement par bloc et sauvegardera le numéro du dernier bloc reçu. De cette façon, s'il reçoit le même bloc de données, le noeud enverra directement l'acquittement sans écrire les données en mémoire.

L'étape suivante vérifiera la validité du fichier reçu en envoyant une signature afin de la comparer avec celle que le noeud calculera. A partir d'ici, si l'on reçoit une erreur du noeud, le processus recommencera du début ou sera stoppé, et s'il reçoit un acquittement alors on considère que la mise à jour est effectuée. Sur le noeud, le module de mise à jour réceptionnera l'ensemble des blocs de données et les sauvegardera dans un espace mémoire réservé. A la réception de la signature, le noeud décompressera le fichier reçu pour vérifier sa signature. Une erreur est alors remonté en cas de fautes détectées, sinon l'adresse de l'application est sauvegardée ainsi que sa signature pour une vérification ultérieure.

Les différentes versions de code auront un numéro qui sera stocké sur le noeud dans la structure de données utilisée pour l'administration. L'utilisation effective est associée au prochain redémarrage du noeud. Dans ces conditions, le noeud doit pouvoir s'éteindre et s'allumer sur demande de l'application de mise à jour. Cette étape interviendra dès que les données reçues seront vérifiées, à la demande du module de mise à jour ou du logiciel d'envoi. Un message au redémarrage du noeud pourra être envoyé pour confirmer la prise en charge correcte de la nouvelle application. Toute erreur de fonctionnement de l'application reçue entraînera le basculement du noeud sur son programme de base et l'envoi d'un message d'erreur.

La gestion des mises à jour est une problématique qui à ce jour ne dispose pas de solution

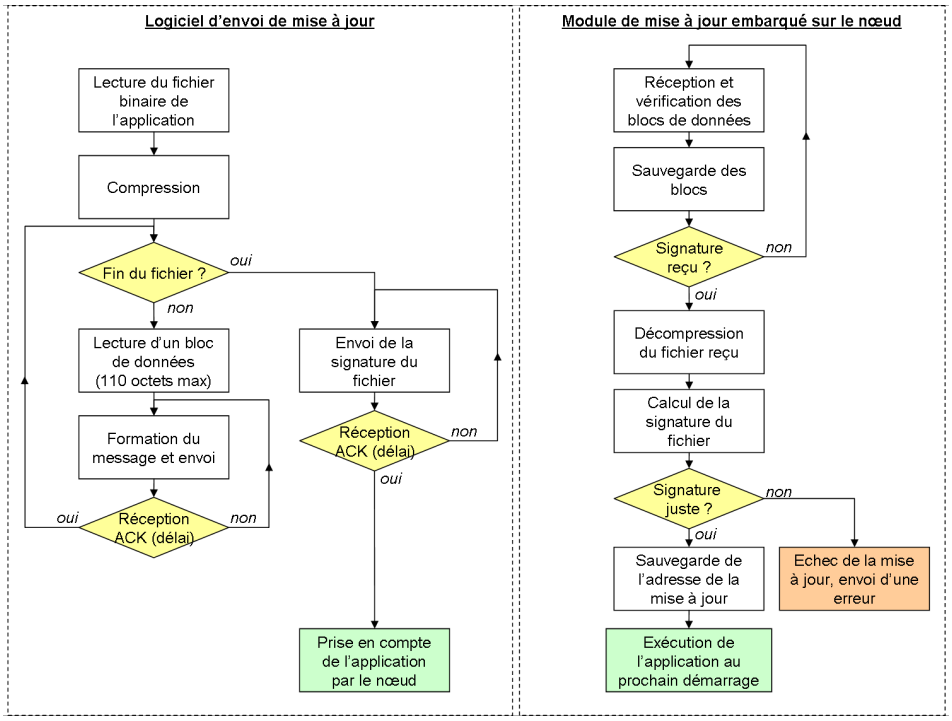

Figure 3.14 – Organigramme des étapes de mise à jour : à gauche la partie serveur, à droite le module intégré aux noeuds

générique. Le choix d'utiliser un logiciel tiers pour diffuser les mises à jour est une solution comme une autre. Sur un réseau vaste contenant de nombreux noeuds, l'envoi des nouvelles applications vers quelques noeuds demandera une méthode de parallélisation pour pouvoir envoyer les blocs de données à plusieurs destinataires. Dans le cas où ce sera une mise à jour générale, la transmission par broadcast pourra être utilisée pour adresser tous les noeuds. Ici, le logiciel de mise à jour devra attendre la réception de l'acquittement de tous les noeuds avant d'envoyer le prochain bloc de données. Le fonctionnement d'une telle solution demande d'avoir des noeuds réactifs afin de perdre le moins de temps possible entre 2 envois.

3.3.2 Diffusion et stockage du nouveau programme

L'exécution d'une nouvelle application commence par la diffusion vers le noeud du fichier qui lui est associé et par son stockage. Pour cela, il est nécessaire de définir une méthode de diffusion efficace et robuste pour permettre la reprise de la diffusion en cas de perte de connexion, et pour garantir les données reçues par le noeud. Enfin, des limites de taille devront être fixées afin de laisser un espace mémoire assez important pour l'utilisateur.

La diffusion des données de la nouvelle application se résume à fournir un mécanisme de fragmentation d'un fichier de données complété par une technique de reconstruction sur le noeud. Pour cela, le protocole LiveNCM-P intègre un champ 'numéro de fragments' dans la définition de ses messages (Figure 3.8). Chaque message peut contenir des paramètres jusqu'à

110 octets ce qui permet dans un premier temps d'envoyer 255 fragments soit une application d'une taille maximale de 28 ko. A titre d'exemple, une application simple de relevé périodique de température avec le module d'administration sans prétraitement des données sera compilée en un fichier binaire de 19 ko. Un espace au moins équivalent à ce volume devra être réservé par l'unité de stockage du noeud. Pour des noeuds ayant peu de ressources mémoire, l'application sera limitée par l'espace de stockage de la mise à jour allouable sur les noeuds. Par contre, pour des noeuds avec d'importantes ressources mémoire, une commande supplémentaire pourra être incluse afin de permettre l'envoi d'applications plus volumineuses. Cette commande se traduira par un décalage de l'index de sauvegarde permettant ainsi la réception de 255 fragments supplémentaires. Par ailleurs, afin de réduire le nombre de fragments, une technique de compression sera appliquée sur le code de l'application à transmettre. Le choix de l'algorithme portera sur ses performances et sur l'utilisation des ressources qu'il nécessite pour s'appliquer. L'impact sur la communication sera bien entendu une contrainte sur ce choix.

Compression des données échangées

Dans les RCSFs, la réduction des coûts énergétiques passe par des techniques réduisant les échanges. Il est donc courant de rencontrer la compression de données pour des volumes de données importants. L'objectif de cette utilisation est de réduire le trafic dans les RCSFs en permettant d'envoyer plus de données en un minimum de messages ou d'envoyer des messages plus courts. Le premier point concerne directement le module de mise à jour. L'idée est de transmettre une application de taille importante en un minimum de messages vers le noeud. Nous avons vu que sans compression une application de 28 ko pouvait être transmise sans nécessité de commande supplémentaire. On peut donc imaginer transmettre plus de données en les compressant. LiveNCM a donc choisi d'implémenter et tester deux algorithmes de compression de type Huffman et LZW connus pour leur simplicité de calcul et leur rapidité d'exécution. L'algorithme de Huffman permet de coder les caractères en fonction de leurs occurrences dans un fichier. Un dictionnaire est alors produit en fonction des résultats de la compression puis transmis vers le destinataire. L'échange du dictionnaire est valable uniquement si le gain de la compression est au moins supérieur à sa taille. Dans les autres cas, la compression de données perd de son intérêt. L'algorithme LZW se base sur un codage de chaînes de caractères récurrentes. Pour chaque nouvelle suite de caractères, un nouveau code est alors produit. Cela impose alors un codage des caractères sur plus de 8 bits par caractère. Dans ce cas, aucun dictionnaire ne sera échangé car le noeud le reconstituera au fur à mesure de la décompression. La diffusion d'une mise à jour est un cas d'utilisation très intéressant pour ces algorithmes. La compression du fichier à envoyer est effectuée par le logiciel en charge de gérer les différentes versions. Un ou plusieurs messages sont alors échangés avec le noeud pour lui permettre ensuite de décompresser les données. Afin d'optimiser l'espace mémoire, la décompression à la volée des différents messages reçus peut être une solution pour reconstruire, sur le noeud, la nouvelle application. A défaut d'utiliser ce mode de décompression, le noeud devra disposer d'un espace mémoire au moins deux fois supérieur à la taille de la mise à jour. Ensuite, une étape de vérification de la mise à jour sera nécessaire afin de valider la prise en charge par le noeud d'une application intègre.

Validation du fichier reçu

Aujourd'hui, le développement d'Internet permet de réaliser des transferts de fichiers depuis des serveurs distants à des vitesses importantes sur des réseaux libres d'accès à tous. Les risques d'erreurs dans la diffusion ou les risques de recevoir une application malsaine ont amené à utiliser des méthodes de vérification robustes des fichiers échangés. Dans ce but, l'algorithme MD5 [Rivest 92] a été développé et utilisé afin de vérifier l'intégrité de données téléchargées pour prévenir les risques de piratage par exemple. L'algorithme MD5 est une fonction de hachage cryptographique qui permet d'obtenir l'empreinte numérique d'un fichier (en l'occurrence une séquence de 128 bits ou 32 caractères en notation hexadécimale) avec une probabilité très forte que deux fichiers différents donnent deux empreintes différentes. Malgré une faille de sécurité pouvant permettre de reconstituer la même empreinte numérique sur deux fichiers différents, l'algorithme MD5 est aujourd'hui présent sur tout fichier ou logiciel de taille importante comme les systèmes d'exploitation libres ou certaines applications importantes. La probabilité d'avoir une collision de MD5 est alors égale à la probabilité de chaque signature. Pour une signature 128 bits, cette probabilité sera de $1/2^{128}$ soit $2.9 * 10^{-39}$. Dans les RCSFs, une telle probabilité constitue un gage de sûreté dans la diffusion de fichiers volumineux. L'empreinte numérique est, dans notre cas, calculée sur le fichier non compressé afin de pouvoir vérifier à la fois la diffusion des messages et la décompression des données sur le noeud. Elle est ensuite diffusée vers le noeud qui procédera à la vérification avec celle qu'il aura calculée. Si les données sont intègres, le noeud autorisera l'utilisation de l'application au prochain redémarrage du noeud en inscrivant l'adresse mémoire de l'application dans la structure d'administration. Dans le cas contraire, un message d'erreur est envoyé vers le logiciel de mise à jour afin de recommencer le processus de diffusion. Les données précédemment reçues sont alors écrasées ou effacées du noeud. Une erreur d'empreinte de fichier conduit forcément au rejet de l'application pour des raisons de sécurité mais aussi pour éliminer tout problème à l'exécution.

Stockage et prise en compte de l'application

Le couple, algorithme de compression et contrôle d'intégrité, permet de garantir les données que le noeud va recevoir. Cependant, une dernière problématique se pose. Le stockage et la prise en charge de l'application sont deux points durs du module de mise à jour. Les intergiciels présentés dans le chapitre 2.2.2 présentent plusieurs pistes pour traduire le fichier reçu en application active. La première consiste à implémenter une machine virtuelle sur le noeud afin de traduire les instructions reçues en actions sur le noeud. Cela implique d'ajouter une interface qui coûte des ressources à la fois mémoire et processeur. La deuxième piste proposée est d'envoyer des modules compilés qui seront intégrés à l'application déjà embarquée sur le noeud par l'intermédiaire d'une interface adaptateur. Ces deux pistes offrent une bonne modularité des programmes embarqués en permettant de modifier une partie des fonctionnalités implémentées. Cependant, l'ajout d'une interface (adaptateur ou machine virtuelle) peut réduire les performances du noeud et donc celles des applications embarquées. La charge de calcul et de mémoire peut devenir importante pour des fichiers d'instructions de grandes tailles.

LiveNCM a fait le choix de mettre à jour l'ensemble de l'application à chaque fois. Le

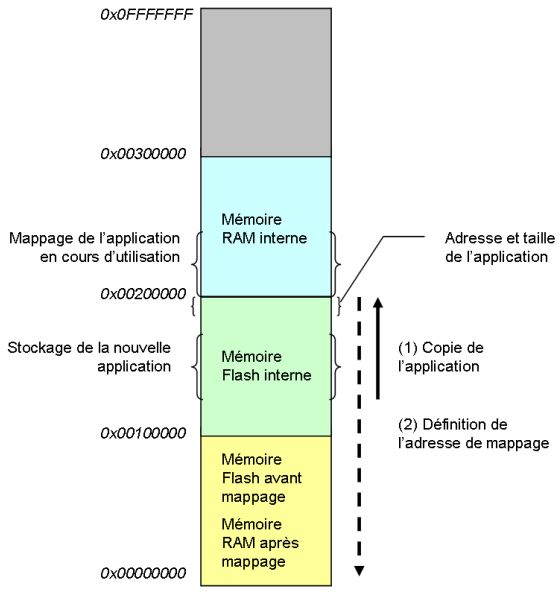

Figure 3.15 – Utilisation de la mémoire par le module de mise à jour

fichier est alors envoyé en binaire afin d'être directement utilisable par le noeud au prochain redémarrage. Pour cela, il sera stocké en mémoire Flash (mémoire interne) avant d'être prise en compte. Le schéma de la Figure 3.15 présente les espaces mémoires mis en oeuvre pour recevoir l'application et la prendre en compte. Avant de pouvoir lancer l'application, trois étapes seront nécessaires au démarrage du noeud. La première consiste à initialiser le fonctionnement du noeud. Ensuite, le module de mise à jour copiera la nouvelle application en mémoire RAM (Figure 3.15(1)) avant de lancer son exécution. La dernière étape consistera à placer le pointeur d'exécution au début de cette zone mémoire (Figure 3.15(2)). La version et l'adresse de stockage de l'application sont sauvegardées en mémoire Flash du processeur. Le noeud n'a plus qu'à lire cette zone mémoire pour savoir s'il y a une mise à jour à utiliser ou s'il doit utiliser le programme de base du noeud.

Avec ce concept de la mise à jour, l'inconvénient majeur est d'obliger l'utilisateur à envoyer l'intégralité de l'application pour une simple correction ou pour l'ajout d'une commande simple. Une solution possible serait d'utiliser des correctifs applicables sur le programme présent sur le noeud. Le fichier reçu ne contiendra que les codes modifiés depuis la version précédente. En choisissant cette technique, le module de mise à jour implémentera un mécanisme pour traiter les correctifs et les appliquer sur le code. La création d'un correctif commencera par le parcours des différences entre la mise à jour précédente et la future. Le fichier résultant décrira des triplets (adresse/type de modification/corrections) qui devront être appliqués. L'adresse du caractère sera le numéro du caractère défini depuis le début du fichier où la modification devra être appliquée. Les opérations possibles seront donc l'insertion, la suppression et la correction de caractères ou de chaîne de caractères. Avec un tel mécanisme, les mises à jour de codes n'utiliseront que peu de messages pour envoyer les modifications vers le noeud mais, par contre, le

temps de prise en compte sera augmenté. Suivant l'opération à appliquer, les traitements seront plus ou moins complexes. Pour limiter l'usure de la mémoire Flash, les modifications seront appliquées sur une version stockée en mémoire RAM sur laquelle il est possible d'effectuer des opérations rapides. La version corrigée viendra ensuite écraser la précédente version sauvegardée sur le noeud après que le résultat ait été validé via le contrôle d'intégrité du module de mise à jour. Les correctifs seront donc principalement utilisés pour des faibles corrections pour lesquelles le noeud peut agir rapidement et facilement. Les plus grosses mises à jour passeront par le premier mécanisme, c'est-à-dire qu'on enverra l'intégralité du code binaire.

3.4 Conclusion

Le besoin d'administration des RCSFs est une question de recherche actuelle. La diffusion, le stockage et le traitement des données par le noeud demande d'avoir de nombreuses méthodes pour assurer ces fonctionnalités. Comme nous avons pu le voir, répondre aux différentes contraintes n'est pas chose facile. Les différentes méthodes et techniques présentées dans le chapitre précédent apportent des méthodes intéressantes pour une meilleure gestion des ressources et des noeuds. Malheureusement, leur aspect intrusif et peu portable les rend difficiles à mettre en oeuvre sur des applications agri-environnementales par exemple. La solution d'administration LiveNCM proposée et présentée par la Figure 3.16 intègre une partie de ces méthodes et propose l'utilisation de techniques innovantes comme le diagnostic indirect et les estimateurs pour limiter son impact sur le fonctionnement général. L'ensemble des messages échangés servent alors de messages d'administration. Les estimateurs viennent en complément pour limiter les données inutilement envoyées et pour aider à prédire à cours terme l'évolution d'une valeur.

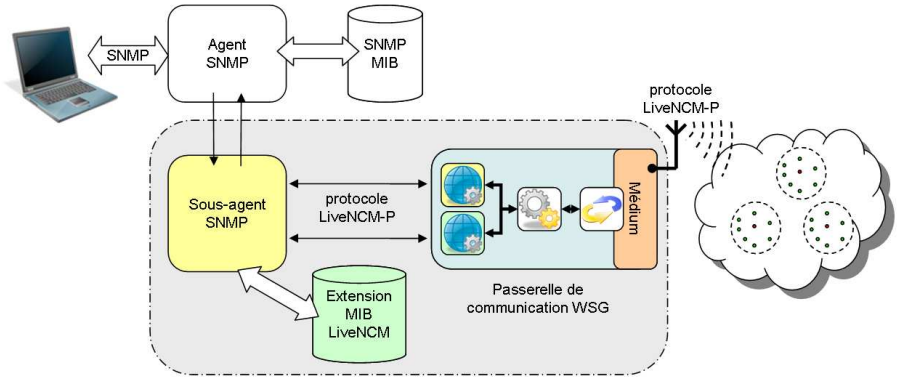

Figure 3.16 – Architecture complète de la solution d'administration LiveNCM

LiveNCM s'appuie sur une architecture multi-agent. La partie fixe avec le sous-agent SNMP et la passerelle de communication WSG constitue une base stable et robuste sur laquelle les différents noeuds du RCSF peuvent compter. Chaque noeud dispose d'un agent embarqué qui s'intègre à l'application sans en influencer son fonctionnement. Il est chargé de la gestion des messages d'administration et du module de mise à jour. La communication est alors confiée au protocole de communication LiveNCM-P qui a été développé pour répondre aux problématiques de minimisation des données échangées et d'économie d'énergie. Il s'appuie pour cela sur une optimisation des entêtes de chaque message et sur des techniques de réduction de données comme les estimateurs et la compression. La Table 3.1 dresse les caractéristiques de LiveNCM face aux autres techniques d'administration utilisant SNMP.

L'architecture d'administration choisie est basée sur une structure hiérarchique en clusters du RCSF. La possibilité d'exécuter des requêtes sur le noeud est présente sur les différents concepts suivant des protocoles de communication divers (SNMP, Propriétaire, LiveNCM-P). Le système de requêtes constitue un atout important pour un accès rapide aux données à

tout instant. Cependant, l'usage qui en est fait, est trop intrusif dans les méthodes vues dans le chapitre précédent. LiveNCM a fait le choix d'introduire le diagnostic indirect pour limiter l'intrusion de l'administration dans le fonctionnement général de l'application. Avec ce concept, il y a moins de requêtes vers le noeud donc une meilleure autonomie.

Tout comme GUERRILLA, LiveNCM intègre des traitements automatisés de certains types de messages afin de la rendre plus flexible et plus réactif face aux différents événements intervenant sur le noeud et dans le RCSF. Les données d'administration sont, dans le cas de LiveNCM, stockées sur un seul et même agent SNMP qui sera employé pour la gestion de plusieurs RCSF comportant plusieurs noeuds et capteurs, contrairement aux autres méthodes où un agent était déployé sur chaque noeud. Certains d'entre eux embarquent des méthodes d'auto-configuration des réseaux ce qui permet de les réorganiser périodiquement. LiveNCM a fait le choix d'externaliser cette partie pour permettre à l'utilisateur d'implanter la méthode de son choix sans qu'il n'y ait à développer une nouvelle version de l'agent embarqué sur les noeuds. L'avantage de procéder de cette manière est de soulager les noeuds de la charge de calcul et de stockage associée à l'agent SNMP. De plus, le format des messages SNMP est inadapté aux RCSFs car sa taille est trop importante. L'inconvénient d'avoir cette structure centralisée est qu'elle introduit un goulot d'étranglement au niveau du sous-agent SNMP ce qui peut le surcharger et augmenter les collisions et les pertes de messages. Enfin, la problématique énergétique est traitée de différentes manières dans chacune de ces méthodes. Si ANMP et SHAMAN ne proposent pas de politique énergétique avancée, il n'en est pas de même pour GUERRILLA et LiveNCM. GUERRILA classe ses noeuds par catégorie de ressources afin de lisser l'énergie consommée. Une réorganisation du réseau est alors effectuée périodiquement pour avoir une rotation des noeuds maîtres. LiveNCM propose une politique énergétique adaptable et basée sur les remontées d'informations des noeuds. Ce module intégré à la passerelle de communication et piloté par les paramètres du sous-agent SNMP, surveille les messages transitant afin de détecter les alertes de pannes ou de dysfonctionnements. Toute alerte détectée par la passerelle se traduit par un message réflexe permettant d'agir sur un ensemble de paramètres du noeud dont le mode de fonctionnement par exemple.

Les solutions prises pour comparaison sont basées sur une architecture similaire à LiveNCM, à savoir l'utilisation du protocole SNMP pour administrer les RCSFs. Nous avons pu voir aussi comment caractériser un protocole d'administration pour ces réseaux. La Table 3.2 tente de présenter ces différents points pour ces solutions d'administration. L'empreinte mémoire étant une contrainte très importante, certaines solutions demandent de disposer de noeuds plus puissants pour soutenir les fonctions d'administration dans les clusters. LiveNCM et GUERRILLA ont la particularité de proposer une réponse prenant en compte cette problématique en intégrant, pour le premier, un développement minimisant l'impact de l'administration sur le fonctionnement du noeud et, pour le second, une architecture tournée vers l'utilisation d'agents mobiles. La robustesse du protocole SNMP n'est plus à prouver, mais une utilisation atypique de celui-ci le rend moins tolérant aux pannes que sur des applications standards.

En résumé, la solution LiveNCM proposée permet de minimiser l'impact de l'administration sur les ressources du noeud tout en apportant une interface générique d'accès aux données basée sur un protocole d'administration reconnu et très diffusé, le protocole SNMP. Son architecture

Table 3.1 – Carte d'identité de la solution d'administration LiveNCM

	ANMP	SHAMAN	GUERRILLA	LiveNCM
Type d'administration	Hiérarchique Sur demande Requêtes SNMP	Hiérarchique Sur demande Scripts SSL	Hiérarchique Hybride Requêtes SNMP et agents mobiles	Hiérarchique Hybride Requêtes spécifiques et actions automatiques
Stockage des données	Décentralisé N agents SNMP	Décentralisé N agents SNMP	Décentralisé N agents SNMP	Centralisé 1 agent SNMP
Protocole de communication	SNMP	Propriétaire	SNMP	LiveNCM-P
Auto-configuration Méthodes utilisées	Clusters Mathématique ou géographique	aucune	Clusters Basé sur les ressources	aucune (Adaptable)
Politique énergétique	Simple	Simple	Avancée Classification des noeuds et rotation des maîtres	Réactive Optimisation des modes de fonctionnement

Table 3.2 – Caractéristiques des solutions d'administration basées sur le protocole SNMP

	ANMP	SHAMAN	GUERRILLA	LiveNCM
Traitements légers	-	-	-	+ +
Robustesse et tolérance aux pannes	+	+	+ +	+ +
Réactivité et adaptabilité	+	+	+ +	+
Modularité	- -	- -	+ +	+
Empreinte mémoire	- -	- -	+	+ +
Intrusion dans le réseau	- -	- -	- -	+ +

est découpée en trois groupes : la partie SNMP, la passerelle de communication et enfin l'agent sur le noeud. Le premier permettra de centraliser les données des RCSFs afin de fournir à l'utilisateur la possibilité de les visualiser à l'aide d'outils SNMP standards. La passerelle de communication WSG vient compléter le protocole SNMP afin de fournir une interface entre le sous-agent et le RCSF. Elle intégrera certaines techniques comme la politique de gestion énergétique ou la synchronisation des noeuds. Elle jouera le rôle de station de base pilotée par le sous-agent SNMP, sans pour autant gérer les données des noeuds. Par ailleurs, la vérification des messages transitant permet de garantir les messages arrivant sur le RCSF. Enfin, sur les noeuds, un agent léger est introduit dans l'application de base afin de traduire les commandes d'administration et de gérer les modes de fonctionnement du noeud. Enfin, les échanges entre les différents noeuds et le sous-agent SNMP ont été formatés suivant le protocole de communication LiveNCM-P. Il a été développé pour respecter les contraintes énergétiques et minimiser les données nécessaires à la diffusion des messages. Ceci se traduit par un formatage des paramètres de chaque message en optimisant le nombre de bits utiles. De plus, la réduction des volumes échangés dans le RCSF passe par deux techniques : l'estimation et la compression. La première doit permettre de diffuser uniquement les mesures utiles pour reconstruire le signal acquis (batterie, température, etc.) avec un seuil d'erreurs paramétrable. La compression de données est principalement employé pour la diffusion de gros volume de données afin de réduire le nombre de messages.

Finalement, LiveNCM offre une nouvelle architecture d'administration intégrant une interface de communication en charge de réaliser des prétraitements sur les messages et d'effectuer une partie de l'administration (tâches périodiques, actions réflexes). La politique énergétique employée par LiveNCM-P doit permettre de répondre rapidement aux contraintes subies par le noeud ainsi qu'aux pannes qu'il peut rencontrer. L'objectif est bien sûr de maximiser la durée de vie du noeud en jouant sur les paramètres importants du noeud. Une évaluation précise des différentes techniques et méthodes intégrées à LiveNCM devra avoir lieu afin de valider leurs performances ainsi que le fonctionnement général de cette nouvelle solution d'administration.

$\sim \star \sim \star \sim \star \sim$

Chapitre 4

Évaluation et implantation de la solution LiveNCM

La solution d'administration LiveNCM présentée dans le chapitre précédent s'appuie sur un ensemble de concepts et techniques existants. L'utilisation du protocole SNMP pour gérer les données des RCSFs a déjà été vue dans des protocoles existants comme ANMP, ou GUERRILLA. Cependant, LiveNCM propose de l'appliquer d'une manière différente en n'utilisant non plus un agent par noeud mais un agent pour tous les réseaux. Une évaluation de cette technique doit être menée pour apprécier les performances d'administration ainsi que les performances des échanges de données. Par ailleurs, le protocole LiveNCM-P utilisé pour les communications s'appuie lui aussi sur des techniques innovantes à évaluer. L'estimation des mesures issues d'un capteur pour réduire le nombre de messages est, à ce jour, une alternative peu coûteuse à la compression de données souvent utilisée. Pour effectuer cette évaluation, une plateforme matérielle a été choisie pour supporter LiveNCM et gérer les contraintes subies par le noeud. La suite de ce chapitre consistera à évaluer les performances des différents éléments de cette solution d'administration avant de présenter un exemple d'application de LiveNCM sur un RCSF collectant des mesures d'humidité du sol pour contrôler l'irrigation des cultures.

4.1 Les plateformes d'évaluation

L'évaluation d'une méthode ou d'une nouvelle technique demande de mettre en place des protocoles stricts afin de pouvoir comparer les différents résultats en se plaçant dans des conditions similaires. En termes de tests et validations, deux phases sont nécessaires. La première consiste à effectuer des simulations, dans des conditions idéales, des différents éléments de la solution LiveNCM. La deuxième se base sur une plateforme matérielle dans un RCSF réel dans le cadre d'une application ou d'une campagne de tests. Pour la première étape, les simulations ont été effectuées sous MatLab 7.0[1]. Pour la seconde partie, les éléments fixes de la solution LiveNCM ont été évalués sur une machine dédiée utilisant un serveur SNMP[2]. L'évaluation de

[1]Ordinateur sous Windows XP équipé d'un processeur 3.4 GHz utilisant 1 Go de mémoire RAM

[2]Ordinateur sous Ubuntu 9.04 équipé d'un processeur 2.8 GHz avec 512 Mo de mémoire RAM

Table 4.1 – Quelques plateformes matérielles existantes pour répondre aux besoins des RCSFs

	LiveNode (2007) [Hou 07b] Univ. Blaise Pascal LIMOS	TmoteSky (2006) [Sentillia 09] Univ. Californie Berkeley	MICAz (2004) [Crossbow 09] Crossbow
			
Processeur	Atmel AT91SAM7S256 RISC 32bits	Texas Instruments MSP430 RISC 16bits	Atmel ATmega128L RISC 8bits
Fréquence	0.5 kHz à 50 MHz	8 MHz	8 MHz
Mémoire Flash Mémoire RAM	256 ko 64 ko	48 ko 10 ko	128 ko 4 ko
Médium sans fil	IEEE 802.15.4	IEEE 802.15.4	IEEE 802.15.4
Portée (m)	300/1600	50/125	30/100
Consommation veille (μA)	50	5.1	16
Consommation Rx/Tx (mA)	55/215	19.5/21.8	27.7/25.4
Capteurs embarqués	au choix	3 capteurs intégrés et 16 ports d'extensions	au choix

la solution LiveNCM embarquée sur une application réelle commencera par le choix d'une plateforme matérielle permettant un contrôle avancé des différents périphériques d'un noeud. Basé sur une optimisation des consommations énergétiques, la caractéristique principale de cette plateforme sera sa capacité à gérer l'énergie utilisée ainsi que les solutions pour la préserver. Dans cette optique, on trouve sur le marché de nombreux systèmes embarqués permettant de réaliser cette évaluation. Dans ce mémoire, nous ne présenterons pas l'ensemble de ces plateformes qui sont décrites dans le livre [García-Hernando 08]. Nous nous concentrerons sur deux plateformes des plus utilisées, MICAz [Crossbow 09] et TmoteSky [Sentillia 09], en comparaison au noeud développé localement, LiveNode [Hou 07b, Jacquot 07b]. La Table 4.1 reprend quelques unes de leurs caractéristiques majeures.

L'évolution des technologies a permis, peu à peu, d'utiliser des processeurs plus performants et moins consommateurs d'énergie. Les capacités de calcul ont donc été multipliées tout comme les capacités mémoire. Même si TmoteSky dispose de moins de mémoire Flash que les autres, la tendance reste bien à l'augmentation des mémoires embarquées afin de réaliser des traitements de plus en plus complexes directement sur les noeuds. Le coût de calcul d'une donnée sur un

noeud est dans la majorité des cas moins important que l'envoi de cette donnée via le médium de communication sans fil. En terme de coût financier, la société Crossbow propose les MICAz aux alentours de 170$ l'unité tandis que la société Sentilla vend le TmoteSky à environ 130$ l'unité. Le coût théorique d'un LiveNode se trouve dans cette fourchette. Par ailleurs, la communication pour ces trois noeuds se base sur une solution ZigBee IEEE 802.15.4 avec des configurations assez différentes. Le LiveNode utilise ce médium dans sa version la plus performante en termes de portée et de débit tandis que pour les deux autres, la puissance d'émission est bien plus faible ce qui implique une baisse de la portée pour ces noeuds. En se basant sur les valeurs des constructeurs, il faudrait disposer d'au moins 13 noeuds relais avec un TmoteSky pour obtenir le même rayon d'action que le module ZigBee employé par le LiveNode mais ceci implique donc une consommation énergétique plus importante des noeuds. Pour finir, les capacités mémoire de ce dernier dépassent celles embarquées sur les deux autres noeuds. Il est alors possible de développer plus d'intelligence sur ces noeuds afin de réduire les communications.

Malgré une consommation un peu plus importante, nous avons choisi d'utiliser la plateforme LiveNode pour effectuer l'évaluation de la solution d'administration LiveNCM. Par ailleurs, le processeur embarqué supporte le changement de fréquence de fonctionnement sur demande ce qui permettra de moduler la consommation énergétique du noeud. Les différents modes de fonctionnement définis par LiveNCM pourront donc s'appuyer sur cette caractéristique et profiter de l'architecture matérielle permettant de contrôler l'alimentation au plus bas niveau. Les LiveNodes intègrent le système "Sleep and WakeUp" s'appuyant sur une horloge temps réel (RTC) pour assurer le réveil des noeuds à heure fixe. Pendant les périodes d'inactivité, le noeud ne consomme rien de plus que le nécessaire pour alimenter cette horloge. La contre partie de ce système est de ne pas pouvoir joindre les noeuds en dehors des plages de communication définies au préalable. Dans un premier temps, les évaluations seront effectuées sans mettre en oeuvre ce système afin de valider l'ensemble des échanges et des modifications. Par la suite, sur l'application que nous présenterons en fin de chapitre, les réveils des noeuds seront fixés par le RTC et les messages seront stockés dans une file d'attente jusqu'au prochain réveil.

4.2 Validation des performances du protocole de communication

Les systèmes informatiques s'appuient sur des protocoles de communication pour échanger des données entre eux. Les RCSFs adoptent la même ligne directrice pour diffuser les informations entre les noeuds et le superviseur. La solution LiveNCM propose alors de fournir une formalisation des messages afin de prendre en compte l'ensemble des contraintes de communication que peuvent subir les noeuds dans leur vie. Le principe de ce protocole est, tout d'abord, de garantir la réception d'un message intègre par le destinataire tout en réduisant l'impact énergétique des transmissions. Ce dernier point demande alors de faire attention aux données inutilement envoyées ainsi qu'à leur format. Pour des valeurs stables, l'intérêt de diffuser toutes les données est minime et consommateur d'énergie. Le protocole LiveNCM-P emploie alors différentes techniques (estimation, compression) pour minimiser l'envoi de données inutiles ou

minimiser les échanges. En complément, ce protocole contrôle les erreurs de diffusion à l'aide d'un CRC afin d'éliminer tout traitement du noeud sur des données erronées.

4.2.1 Réduire les données échangées, appréciation des techniques portées par LiveNCM

Dans les RCSFs, la ressource à optimiser reste toujours la source énergétique. L'épuiser revient à condamner le noeud au silence avec les conséquences qui en découlent : perte des mesures, perte de connectivité vers des noeuds ou vers une plus large partie du réseau (noeud relais). De telles conséquences doivent être évitées au maximum pour préserver et respecter l'autonomie des noeuds et du réseau. La consommation énergétique du noeud est directement liée à l'utilisation du médium d'accès sans fil. La réduction des échanges aura donc pour effet de réduire la consommation des noeuds qu'ils soient esclaves, relais ou maîtres. Le deuxième effet de la minimisation des échanges est de permettre une meilleure fluidité dans le réseau en réduisant les collisions et les goulots d'étranglement. Pour répondre à la question de réduction des échanges, la solution d'administration LiveNCM prévoit l'utilisation d'un estimateur pour les mesures périodiques et l'intégration d'un algorithme de compression de type Huffman ou LZW pour la diffusion des mises à jour vers les noeuds.

Évaluation de l'estimateur embarqué

La première technique présente dans LiveNCM se base sur une méthode d'estimation appliquée sur les mesures relevées. L'objectif premier est de limiter les échanges entre le noeud et son superviseur en éliminant les données non déterminantes en fonction d'une marge d'erreur fixée par l'utilisateur. La définition de l'écart entre la mesure et la valeur estimée se basera sur la précision du capteur. Cependant, il est possible de la fixer afin d'agir plus fortement sur les envois. L'objectif secondaire est de fournir à l'utilisateur la tendance de la mesure. En s'appuyant sur un modèle d'estimation utilisant un polynôme d'ordre 1, l'utilisateur disposera de valeurs estimées entre deux envois du noeud. Le modèle très simple se base sur une succession de droites d'estimation pour reconstruire la grandeur observée. L'évaluation de ce mécanisme doit s'effectuer en plusieurs étapes. La première étape sera de chiffrer les gains théoriques sur des courbes générées à différentes fréquences accompagnées de plus ou moins de bruit. La seconde étape sera de comparer les résultats précédents en appliquant l'estimation sur un ensemble de courbes réelles (température air-eau, humidité, vitesse du vent, débit, etc.). A l'issue de cette étape, on définira le type d'observations visées par cette méthode. La dernière étape évaluera les gains énergétiques de la réduction des échanges.

Avant de commencer l'évaluation du modèle d'estimation choisi, nous avons étudié les différents estimateurs possibles. Notre choix s'étant porté sur des estimateurs simples et facile à mettre en oeuvre du type polynomial, une première comparaison doit être effectuée pour voir les avantages des différents modèles polynomiales de l'ordre 0 à l'ordre 3. Le protocole de test est alors simple, un signal sinusoïdal est appliqué en entrée des différents modèles et, en sortie, on obtient les performances de chacun sur un signal commun. La Table 4.2 reprend les résultats

Table 4.2 – Résumé des performances des estimateurs sur un signal sinusoïdal de fréquence 1Hz avec un seuil de 0.3V

Modèle d'estimation	Nombre de valeurs envoyées	Erreur quadratique moyenne	Consommation énergétique du médium
Polynôme d'ordre 0	124 (12.4%)	0.026	9.3%
Polynôme d'ordre 1	28 (2.8%)	0.022	2.1%
Polynôme d'ordre 1 avec synchronisation (60ms)	31 (3.1%)	0.020	2.3%
Polynôme d'ordre 2	607 (60.7%)	0.027	45.5%
Polynôme d'ordre 3	596 (59.6%)	0.019	44.7%
Aucun	1000 (100%)	0	75%

obtenus en appliquant les différents estimateurs sur un signal sinusoïdal lent de fréquence 1Hz pour un seuil de 0.3V avec une fréquence de synchronisation pour l'estimateur de LiveNCM de 60 échantillons. La précision de l'estimateur est donnée par la colonne présentant l'Erreur Quadratique Moyenne (EQM), et une estimation linéaire simple du gain énergétique est donnée à titre d'information. Le premier constat est qu'en appliquant un estimateur simple, il est possible de réduire le volume de données échangées d'au moins 50%. Le gain énergétique est alors d'au moins 60% par rapport à l'envoi de toutes les données. Les estimateurs à base d'un polynôme d'ordre supérieur à 1 nécessitent de nombreux échanges pour mettre à jour et suivre l'évolution du signal observé. Avec 60% des échantillons envoyés sur ce signal, la précision du signal reconstruit par un polynôme de degré 3 est la meilleure car l'EQM est la plus faible. Un modèle d'ordre 2 n'offre pas, pour ce signal, une estimation précise. Avec un modèle polynomial d'ordre 0, le signal est reconstruit par une suite d'échelons de taille variable définie par la marge d'erreur. Le volume de données est alors assez faible avec seulement 12.4% des échantillons. Avec un modèle aussi simple et peu de points, la précision de la courbe estimée n'est pas bonne car son EQM est la plus mauvaise. Enfin, un estimateur polynomial d'ordre 1 présente des performances intéressantes. Cet estimateur a une précision proche d'un modèle polynomiale du 3ème ordre en utilisant 20 fois moins d'échantillons. Par ailleurs, en diminuant de 97% les échanges de mesures, le gain énergétique associé est alors très important (-98%).

Avec de telles performances, les estimateurs constituent une technique très efficace pour réduire les transmissions de données dans les RCSFs. Pour valider ces résultats, une deuxième évaluation est nécessaire sur des grandeurs réelles telles que le suivi de température, d'humidité, de la vitesse du vent, des précipitations et du débit d'un cours d'eau. Pour cela, nous nous sommes basés sur des relevés de stations météorologiques surveillées par le Cemagref. La Table 4.3 présente les résultats des estimateurs sur ces différentes grandeurs (Figure 4.1). Le gain en terme d'échantillons échangés varie de 3% jusqu'à 93%. Chaque grandeur dispose d'une mesure par 1h sur des périodes allant de 3 à 12 mois (3 mois pour la température, l'humidité et la vitesse du vent, et 12 mois pour le débit, les précipitations et la hauteur d'eau). On remarque alors que les estimateurs sont particulièrement efficaces pour réduire les informations liées aux

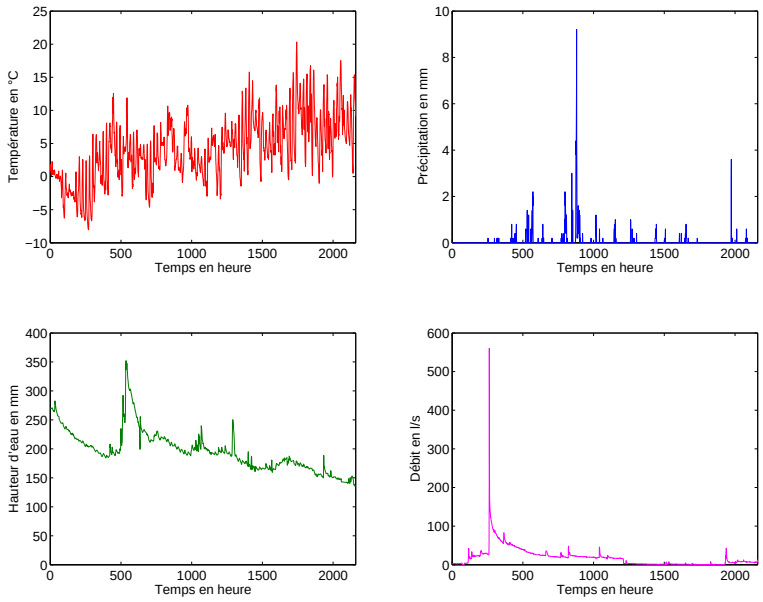

Figure 4.1 – Exemples de relevés effectués par les stations météorologiques du Cemagref (température, précipitation, hauteur d'eau, débit)

précipitations. Avec un gain de 93% avec un modèle polynomial d'ordre 0, cet estimateur est le plus performant car les données évoluaient déjà en échelon. LiveNCM réduit de 88% le nombre d'échantillons en utilisant un modèle polynomial d'ordre 1. En se fixant la marge d'erreur à dix fois la résolution du capteur, la précision pour chaque grandeur est alors très élevée car l'EQM est très faible. Pour finir, en prenant en compte les gains et l'EQM des différents estimateurs, celui qui est le plus économe en terme de mesures échangés reste le modèle polynomial d'ordre 1 avec ou sans synchronisation. Le plus précis est bien sûr basé sur un polynôme du troisième ordre.

Par ailleurs, en augmentant le seuil de tolérance des estimateurs, cette tendance reste toujours vraie. Le gain apporté par l'estimateur d'ordre 1 s'accroît tandis que la précision du modèle de troisième ordre reste la plus importante. LiveNCM implémente donc un estimateur qui permet d'avoir une grande plage d'utilisation tout en gardant une précision correcte. Bien entendu, plus le nombre de points sera élevé, plus le gain apporté par les estimateurs sera élevé car les variations entre deux mesures seront plus faibles.

Maintenant que l'on connaît les performances de l'estimateur employé avec LiveNCM, il est nécessaire d'évaluer les limites de cette méthode concernant la fréquence des signaux observables. En effet, avec des signaux lents, il est facile de trouver des sections linéaires dans le temps. Le calcul de l'estimation reste faisable et il y a une réduction effective des données. Pour des signaux plus rapides, le taux d'échantillonnage du signal jouera un rôle très important. Pour cela, le respect du théorème de Nyquist Shannon (la fréquence de mesure doit être

Table 4.3 – Résultats des différents estimateurs sur des grandeurs physiques

Type de grandeur	Modèle d'estimation	Echantillons acquis	Echantillons utiles	Pourcentage de points utilsés	Erreur quadratique moyenne	Conditions
Température	Polynome ordre 0	2136	1742	81,55	0,0005	nom : th1
	Polynome ordre 1	2136	1657	77,57	0,0007	seuil 0,1 °C
	Polynome ordre 1 synchronisé	2136	1657	77,57	0,0007	synchronisation 1/24h
	Polynome ordre 2	2136	1814	84,93	0,0004	
	Polynome ordre 3	2136	1954	91,48	0,0003	
Vent	Polynome ordre 0	2136	1801	84,32	0,0005	nom : vh1
	Polynome ordre 1	2136	1908	89,33	0,0003	seuil 0,1 m/s
	Polynome ordre 1 synchronisé	2136	1908	89,33	0,0003	synchronisation 1/24h
	Polynome ordre 2	2136	2002	93,73	0,0002	
	Polynome ordre 3	2136	2034	95,22	0,0001	
Humidité	Polynome ordre 0	2136	1965	91,99	0,0002	nom : uh1
	Polynome ordre 1	2136	1959	91,71	0,00025	seuil 0,1 %
	Polynome ordre 1 synchronisé	2136	1958	91,67	0,0003	synchronisation 1/24h
	Polynome ordre 2	2136	2033	95,18	0,00015	
	Polynome ordre 3	2136	2070	96,91	0,0001	
Précipitations	Polynome ordre 0	8759	635	7,25	0	nom : ph1
	Polynome ordre 1	8759	796	9,09	0,000025	seuil 0,1 mm
	Polynome ordre 1 synchronisé	8759	1080	12,33	0,000006	synchronisation 1/24h
	Polynome ordre 2	8759	1080	12,33	0,0001	
	Polynome ordre 3	8759	3888	44,39	0,0012	
Débit	Polynome ordre 0	8760	1534	17,51	0,051	nom : qh1
	Polynome ordre 1	8760	1097	12,52	0,041	seuil 0,5 l/s
	Polynome ordre 1 synchronisé	8760	1252	14,29	0,031	synchronisation 1/24h
	Polynome ordre 2	8760	5513	62,93	0,047	
	Polynome ordre 3	8760	6426	73,36	0,03	
Hauteur	Polynome ordre 0	8700	6263	71,99	0,00000002	nom : hh1
	Polynome ordre 1	8700	3767	43,30	0,00002	seuil 0,1 mm
	Polynome ordre 1 synchronisé	8700	3855	44,31	0,00002	synchronisation 1/24h
	Polynome ordre 2	8700	5587	64,22	0,00094	
	Polynome ordre 3	8700	7320	84,14	0,00096	

Modèle le plus performant
Modèle le plus précis

au moins 2 fois supérieure à celle du signal) est très important. Pour un signal de 100Hz par exemple, on produira des échantillons à 200Hz ce qui devient très important d'un point de vue communication. Dans ce cas de figure, les estimateurs sont attendus pour réduire fortement les échanges afin de réduire l'engorgement du réseau et ainsi éviter le blocage de tout échange entre les noeuds avec le superviseur. Le protocole de test prévoit donc d'évaluer des signaux sinusoïdaux de fréquences variant de 10Hz à 500Hz d'amplitude 10V qui sont échantillonnés à des fréquences variant de 2 à 50 fois la fréquence du signal en entrée. Les graphes de la Figure 4.2 présentent les différents résultats de l'estimateur sur un signal de fréquence 50Hz échantillonné de 100Hz à 2500Hz avec un seuil de 0.2V sans synchronisation. La droite représente le nombre d'échantillons produits tandis que la courbe présente les points utilisés pour reproduire le signal. On remarque alors qu'en respectant le théorème de Nyquist Shannon, l'apport de l'estimateur est nul dans un premier temps avant d'augmenter quand on utilise des fréquences d'échantillonnage supérieures à 12 fois celle du signal observé. Pour des relevés moins fréquents, l'estimateur ne peut que fournir toutes les mesures, il n'y aura aucun gain et l'EQM sera nulle ce qui revient à avoir une précision maximale. Par contre pour des fréquences supérieures, il est possible de réduire les valeurs envoyées. La discrétisation du signal observé joue donc un rôle important pour l'utilisation de l'estimateur à haute fréquence. Ce sera donc une première condition d'utilisation des estimateurs. Pour des signaux de fréquences supérieures à quelques dizaines d'hertz, l'échantillonnage du signal devra être au moins 12 fois supérieur à la fréquence du signal observé. Pour vérifier la validité de ce résultat, une expérimentation en pratique a été menée sur deux signaux générés aux fréquences 1Hz et 10Hz, avec un offset de 5.6V et une amplitude de 3.5V. La discrétisation du signal analogique a été effectuée à 100Hz pour ces deux signaux. Les relevés montrent que pour le signal basse fréquence (1Hz), un gain de 80% des points est obtenu pour un seuil fixé à 0.2V. Popur le signal à 10Hz, le gain observée est alors de 10% avec le même paramétrage. Ces même signaux ont été générés sous MatLab pour vérifier ces observations. Les résultats théoriques obtenus par simulation corroborent les relevés effectuées en pratique.

En définitive, l'estimateur polynomial d'ordre 1 intégré à LiveNCM permet d'obtenir de bonnes performances sur des signaux à faible fréquence quel que soit l'échantillonnage du signal. Pour des relevés environnementaux, où les variations sont lentes, un estimateur de ce type amène un gain pour la communication. Pour ceux ayant des fréquences plus élevées, il est nécessaire de réaliser plus de mesures pour reconstruire le signal mais aussi pour que l'estimateur apporte un gain en messages. Le seuil de tolérance sera fixé par l'utilisateur afin de trouver un compromis entre précision et autonomie. Au cours du temps, une modification de cette valeur pourra permettre à l'utilisateur de prolonger la vie du noeud.

Validation de la compression de données utile au module de mise à jour

Après avoir vu comment réduire les données périodiques, il convient d'évoquer comment stocker et diffuser des volumes de données plus importants en un minimum d'espace mémoire ou de messages. Pour cela, LiveNCM compare deux algorithmes de compression simples basés sur les méthodes de Huffman et LZW. La technique de Huffman permet de définir un codage variable pour les caractères utilisés en fonction de leur nombre d'occurrences dans le fichier

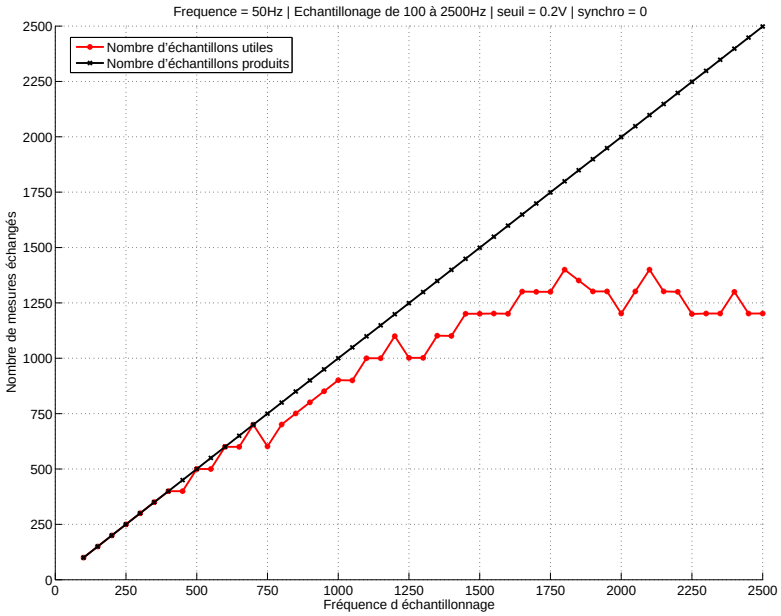

Figure 4.2 – Impact de l'échantillonnage sur l'estimateur polynomial d'ordre 1 avec un signal sinusoïdal de 50Hz avec un seuil fixé à 0.2V

à envoyer. L'inconvénient est qu'il est nécessaire de diffuser ce codage avant de procéder à la décompression des données par le destinataire. Pour pouvoir utiliser cette technique, il faudra que le gain de la compression soit au moins supérieur à la taille du dictionnaire. Une première évaluation de la compression est nécessaire pour déterminer ces performances minimales. Pour cette étape, l'algorithme de Huffman a été implémenté sous MatLab pour pouvoir simuler un grand nombre de messages aléatoires afin d'apprécier le taux de compression. L'exercice consiste à produire 1000 messages aléatoires de taille maximale (127 octets) sur lesquels est appliqué l'algorithme de Huffman. Seul le champ de données du message sera compressé soit 110 octets. Les différents résultats montrent que le taux de compression minimal obtenu se situe autour de 84%. Il est donc possible de gagner 15% d'espace en appliquant l'algorithme de Huffman sur des données aléatoires.

Dans le fonctionnement général de la solution LiveNCM, la compression de données est principalement utilisée pour la diffusion des mises à jour et pour leur stockage sur le noeud. L'impact de la compression sur ces deux points doit donc être apprécié afin de comprendre l'intérêt d'utiliser un tel algorithme. Pour cela, nous avons produit 3 applications simples qui ont été compilées. Les fichiers binaires des applications sont alors utilisés par MatLab. La première étape consiste alors à appliquer la compression avant de découper le résultat en 'n' messages. Tous les blocs de données sont alors stockés avant de procéder à leur décompression. Pour cette étape, on utilise une décompression sur la totalité du fichier. Sur les 3 fichiers binaires, la compression n'apporte finalement que 18% de gain. La structure des fichiers binaires résultant de la compilation d'une application est très variée. Une compression de type Huffman n'obtient

Table 4.4 – Performances des algorithmes de compression appliqués sur une application de 24 kilo-octets

Algorithme	Taille du fichier	Taux de compression	Temps de compression (décompression)
Huffman	19775 octets	82.6%	593ms (3660ms)
LZW	12527 octets	52.3%	448184ms (2375ms)

alors qu'un faible taux de compression de l'ordre de 17%. Ce gain, certes faible, permet de réduire tout de même le volume de données échangées. Les résultats obtenus montrent qu'un algorithme aussi simple que celui de Huffman ne suffit pas à avoir de bonnes performances sur des fichiers avec des caractères à faibles occurences. C'est pourquoi l'algorithme de Huffman est couplé à la méthode LZ77 au sein de la méthode DEFLATE. Un autre algorithme comme LZW a donc été testé en simulation pour pouvoir comparer ces deux algorithmes. Sur une même application si Huffman apporte un gain de 18%, l'algorithme LZW offre une compression plus importante de l'ordre de 48%. Cependant, les temps de calcul sont bien plus élevés pour la seconde méthode par rapport à la première. La Table 4.4 permet de comparer les performances de ces deux algorithmes sur une mise à jour envoyées aux noeuds. On remarque que les temps de compression sont très différents avec un facteur de presque 1000 entre elles. Par contre la décompression des données est bien plus rapide pour l'algorithme LZW que pour la méthode de Huffman. Pour une utilisation dédiée au module de mise à jour, l'algorithme LZW semble être mieux adapté pour répondre à cette problématique. L'ordinateur supportant le logiciel de diffusion des nouvelles applications disposera d'assez de ressources pour exécuter l'algorithme rapidement. Sur le noeud, le temps de décompression est un facteur déterminant pour le choix d'une méthode. L'algorithme LZW reconstitue les données plus rapidement que l'algorithme de Huffman sans nécessiter la diffusion d'un dictionnaire.

En résumé, la compression de données permet évidement de réduire les données échangées. Cependant, le coût de cette réduction se traduit par une augmentation significative des temps de calculs. L'application de différentes compressions sur des blocs de données de taille maximum de 110 octets n'offrira alors aucun gain. Pour l'algorithme de Huffman, il sera nécessaire de produire N dictionnaires pour N messages et pour l'algorithme LZW une même séquence de données peut apparaître plusieurs fois dans un autre bloc de données. En ayant une vue globale du fichier, la compression sera meilleure car les algorithmes auront plus d'informations à leur disposition pour pouvoir créer leurs dictionnaires.

Compression ou estimation, un choix important dans la vie du noeud

L'étude de ces deux techniques a permis de mettre en évidence des avantages et inconvénients ce qui se traduit par des performances plus ou moins bonnes. Le gain d'un estimateur est directement lié à l'échantillonnage du signal et au seuil que l'utilisateur définira. La compression obtiendra des performances très différentes selon le type de données en entrée. Il est donc

nécessaire de voir comment faire une choix entre ces deux méthodes selon les données à échanger. Si dans un premier temps, on cantonnera la compression de données au module de mises à jour et l'estimation aux données périodiques, il serait intéressant de combiner les deux techniques ou de diversifier leur utilisation. La compression de données est souvent utilisée comme une fonctionnalité liée à un système de fichiers implanté sur les noeuds. L'objectif reste le même, l'espace mémoire étant limité, la compression permet de stocker plus de données en un minimum de place. Dans LiveNCM, les mises à jour reçues qui sont actuellement décompressées juste après la réception, pourrait n'être décompressées qu'au lancement de l'application sur le noeud. Par ailleurs, sur chaque noeud, une structure d'administration est sauvegardée à chaque arrêt du noeud. En la compressant, l'utilisateur pourra enregistrer plus d'informations sur l'état du noeud en utilisant le même espace mémoire. Le rôle de l'estimateur a été défini pour réduire les données échangées dans le futur. Pour cela, une hypothèse a été posée : le signal évoluera sur un cours instant de façon linéaire. Avec une telle hypothèse, on fait un pari sur l'avenir d'un signal. Par exemple, la décharge d'une batterie peut suivre une loi linéaire. Avec un estimateur, il est alors possible de prévoir les valeurs futures de la batterie pour anticiper le passage des seuils critiques. Le noeud aura la capacité de prendre une décision avant même que le seuil critique ne soit atteint et donc avant de voir ses performances décroître. En résumé, même si LiveNCM définit un rôle précis à chacune de ces techniques, leur champ d'applications n'est pas limité, pour autant. La politique énergétique déployée dans cette solution prendra en compte ces deux techniques pour prédire les futures valeurs de paramètres d'un noeud dans le cadre de prises de décisions, ou pour réduire les volumes de données en fonction des ressources restantes du noeud. Elle agrégera les différentes mesures sur une période avant de les compresser et de les envoyer vers le noeud maître ou le superviseur.

4.2.2 Déploiement de la passerelle WSG

La solution LiveNCM s'appuie sur une interface externe, la passerelle WSG, afin de lier le RCSF et le sous-agent SNMP. Son rôle est donc de transférer les requêtes et les réponses de l'un vers l'autre en appliquant éventuellement quelques traitements de données. Les interfaces de communication entre le sous-agent SNMP et la passerelle ont été choisies afin de permettre une interaction à distance. L'intérêt est de permettre un déploiement sur deux sites distants interconnectés par un réseau informatique commun. Typiquement, une connexion à Internet ou un réseau privé sont utilisés pour dialoguer entre ces deux éléments. Une évaluation des performances de ce lien est impossible à fournir tant les conditions et la charge du médium de communication sont difficiles à évaluer. Dans cette partie, on se chargera de comprendre le fonctionnement technique de la passerelle tout en essayant d'évaluer les délais qu'elle introduit dans les échanges de requêtes.

L'objectif premier est donc de fournir un service de transfert de données d'un médium vers un autre. Pour cela, il a été nécessaire de développer une application qui réalisera cette fonction. L'architecture idéale est alors de proposer une application multitâche pour traiter les données reçues depuis les noeuds et le sous-agent. Au lancement, l'application crée un ensemble de processus connectés entre eux pour gérer chaque fonctionnalité : l'interface avec le sous-

agent, l'interface avec le RCSF, et les prétraitements. Malheureusement, la portabilité de cette application est très basse voir nulle si l'on souhaite l'utiliser sur des noeuds évolués et intégrés au réseau. Un système d'exploitation devra donc être utilisé pour obtenir le même fonctionnement sur un noeud ou sur un ordinateur. Pour favoriser la généricité de l'application, il est préférable de réduire ses performances en choisissant un traitement séquentiel. De cette manière, il sera très simple d'utiliser la même base logicielle sur un ordinateur et sur un noeud évolué. Les différentes briques logicielles développées seront ainsi partagées entre ces deux plateformes afin de rendre l'adaptation la plus simple possible. Seule la gestion du matériel sera à implémenter pour chaque plateforme. Les fonctions gardant les mêmes paramètres et les mêmes prototypes, il ne sera pas nécessaire de modifier le reste de l'application en changeant de plateforme. Le langage utilisé, pour être le plus portable possible, est le langage C employé aussi bien sur des systèmes Linux ou Windows que sur les noeuds. Avec un compilateur plus performant pour les noeuds, il serait tout à fait possible d'utiliser le langage C++ qui permettrait de développer un noyau dur de l'administration auquel viendraient se greffer les classes spécifiques à la plateforme. L'utilisateur n'aurait alors qu'à surcharger certaines fonctions (envoi et réception de données) pour changer de plateforme. En résumé, l'architecture choisie est alors séquentielle, c'est-à-dire que les messages sont traités les uns après les autres. Pour éviter de perdre des messages, un ordonnanceur de message est inclus à la passerelle pour mettre en file d'attente les messages. Il est basé uniquement sur les priorités des messages, c'est-à-dire que le plus prioritaire est traité à chaque itération. Les performances d'un ordonnanceur de ce type sont déjà bonnes tant qu'un message ne reste pas indéfiniment dans la file d'attente. Pour corriger cette défaillance, l'utilisation de la date d'arrivée du message peut constituer un second critère de choix d'ordre de traitement.

Pour réaliser sa mission, la passerelle WSG se base sur des interfaces de type TCP et UDP pour recevoir et envoyer les messages vers le sous-agent. Par ailleurs, l'accès au médium d'accès sans fil (IEEE 802.15.4) s'effectue sur un ordinateur via une liaison série. L'application aura donc 3 interfaces à surveiller en permanence. Le mode de fonctionnement séquentiel implique de traiter ces périphériques les uns après les autres sans rester bloqué sur l'une d'entre elles. Pour cela, une fonction bloquante surveille ces interfaces simultanément. A chaque caractère reçu, elle lancera le traitement associé au périphérique après quoi elle traitera l'hypothétique message dans le buffer associé. Le fonctionnement sur un noeud est bien différent. En effet, l'utilisation des interruptions permet une meilleure réactivité du noeud et donc un traitement plus rapide des messages. Le caractère reçu est directement inséré dans le buffer de réception sans perturber les traitements en cours. Pour stocker les informations reçues, nous avons opté pour des buffers circulaires faciles à gérer. Avec ce mécanisme, les nouveaux caractères reçus n'écrasent pas directement les précédents ce qui permet un léger délai dans le traitement des messages. Cependant, la contrainte mémoire des noeuds imposent de fixer une taille limite pour laisser de l'espace mémoire à l'application. Les délais de traitement des messages devront donc être optimisés pour minimiser le temps d'attente d'un message dans le buffer. Sur un ordinateur, cette contrainte n'existe pas car on a en général la possibilité d'augmenter la taille des buffers pour rattraper les délais de traitement.

En résumé, la passerelle WSG est une application séquentielle utilisant un maximum de

codes communs entre un noeud et un ordinateur. L'évaluation des performances d'une telle structure demande de se placer dans des conditions spécifiques de trafic et de charge de l'hôte. Les temps de détection des messages, évalués à en moyenne 0.05ms sous MatLab, n'influenceront pas les limitations de la passerelle. Les différentes sources de limitations viendront du médium de communication, des éventuels traitements sur les messages et dans le cas d'une implémentation sur des noeuds, des ressources matérielles disponibles. Le fonctionnement de la passerelle se base sur l'utilisation d'un port UDP commun et d'un port TCP propre à chacune. Cette caractéristique limitera donc le nombre de passerelles possibles sur un même ordinateur.

4.3 Mise à jour à distance des noeuds, performances théoriques

Le module de mise à jour à distance proposé par LiveNCM se base sur un découpage en plusieurs messages d'un fichier binaire compressé représentant l'application choisie. L'évaluation d'un tel module commence donc par l'évaluation de la compression de données. La section 4.2.1.0 a permis de mettre en avant les performances des deux techniques de compression envisagées pour ce module. Les temps de compression certes importants dans certains cas ne poseront pas de problèmes au bon fonctionnement de la solution car elle s'effectuera sur une machine disposant de toutes les ressources nécessaires. Le temps de décompression est, par contre, plus important car il représente le temps que le noeud va utiliser pour reconstituer l'application avant de la prendre en compte. Par ailleurs, le temps de diffusion des données est associé à la configuration du médium de communication. La vitesse de transferts (baudrate) varie généralement de 9600 bauds (1.2 ko/s) à 115200 bauds (14.4 ko/s) ce qui permet d'avoir de meilleurs débits en diffusion de données. La contrepartie est une augmentation des erreurs de transmissions. C'est pourquoi, nous avons choisi un baudrate faible de 9600 bauds pour l'ensemble des liaisons qu'elles soient sans fil (ZigBee, Wi-Fi) ou filaires (liaison série). La Table 4.5 présente les performances théoriques du module de mise à jour avec cette configuration. La première ligne correspond à une application standard de relevé périodique de température, tandis que les deux suivantes représentent des applications de taille maximale avec ou sans extension de la mémoire. Les 3 premières lignes concernent des applications envoyées sans compression tandis que les 3 dernières représentent l'envoi de ces mêmes applications après compression par l'algorithme LZW. Le temps de mise à jour doit alors tenir compte du temps de décompression du fichier reçu. Les évaluations sont effectuées sur l'ordinateur de test sous MatLab. Les performances de décompression seront donc meilleures que si l'on utilisait des noeuds pour effectuer cette opération. Les résultats de cette table mettent en évidence l'intérêt d'utiliser un algorithme de compression pour l'échange de mise à jour vers les noeuds. Une application lourde et complexe pesant plus de 56 ko qui sera compressée avant l'envoi, utilisera le même temps de transfert que pour la diffusion d'une application simple (18.9 ko) de relevé de température.

Le module de mise à jour propose des performances relativement intéressantes lorsqu'il n'y a que quelques noeuds. Pour un réseau plus vaste, il sera nécessaire de mettre en place un méca-

Table 4.5 – Temps de mise à jour d'un noeud pour 3 applications

Taille du fichier	Nombre de fragments de données	Volume de données échangées	Temps total de mise à jour
Sans compression			
18900 octets	172	24798 octets (131%)	20.68s
28050 octets	255	36770 octets (131%)	35.66s
56100 octets	510	73490 octets (131%)	61.26s
Avec compression (LZW) (Temps de décompression en plus)			
9975 octets	91	13119 octets (132%)	10.95s (+1.65s)
13201 octets	121	17365 octets (132%)	14.85s (+2.57s)
21971 octets	200	28821 octets (131%)	24.03s (+8.51s)

nisme de diffusion entre les noeuds des mises à jour ou d'adapter le logiciel afin qu'il réalise cette opération sur plusieurs noeuds en parallèle. Une dernière solution consiste à diffuser la nouvelle application à la passerelle de communication WSG qui la diffusera vers les noeuds concernés. Le fichier reçu sera alors compressé pour limiter les traitements intermédiaires supplémentaires. Les acquittements reçus pour chaque bloc de données pourront servir pour débloquer l'envoi suivant par exemple. De plus, elle pourra réguler les mises à jour des noeuds afin de ne pas surcharger le réseau et donc perdre des informations importantes remontant des noeuds comme les mesures ou les alertes.

4.4 Impact énergétique du volume de données échangées sur l'autonomie des noeuds

Après avoir évalué les grandes parties du protocole LiveNCM-P, nous allons maintenant voir qu'elle sera l'influence de l'estimateur sur des volumes plus importants de données avec un RCSF comportant plusieurs milliers de noeuds. Pour cela, nous avons implémenté une application MatLab qui aura pour fonction d'exploiter des grandeurs réelles relevées par des stations du Cemagref, pour les répartir sur un ensemble de noeuds. La simulation porte sur un réseau de 10000 noeuds exploitant 12 grandeurs différentes, 3 relevés de précipitations, 3 relevés de température, 3 hauteurs d'eau, 1 relevé d'humidité ambiante, 1 relevé de débit d'eau, et enfin 1 relevé de vitesse du vent. Les relevés sont effectués sur 3 mois avec une fréquence d'une fois par heure, ce qui représente 2160 échantillons par capteur. Chaque noeud comporte un nombre aléatoire de capteurs allant de 1 à 12 pouvant utiliser plusieurs fois un même relevé de données réelles. Par ailleurs, on considère pour cette simulation qu'il n'y a aucune perte de messages et donc pas de retransmissions. Sur l'ensemble du réseau, 62406 capteurs ont été répartis aléatoirement représentant une moyenne de 6.24 capteurs par noeud avec un écart type de 3.32. On exploite donc l'ensemble des possibilités de capteurs sur les noeuds du réseau. Le volume de

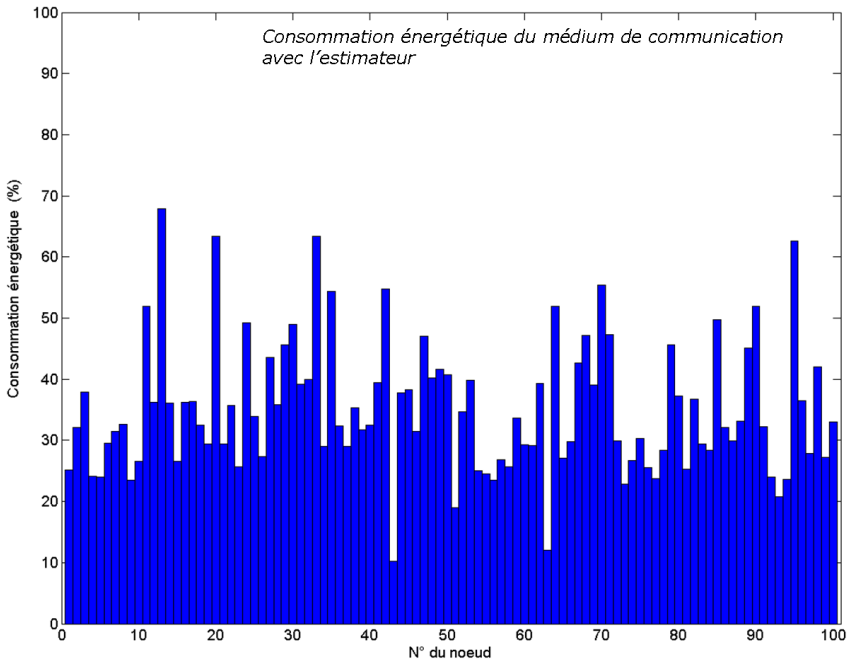

Figure 4.3 – Consommation énergétique du médium de communication avec l'utilisation d'un estimateur

données sans estimateur s'élève à 1.5 Go sur un réseau de ce type. Avec l'estimateur polynomial d'ordre 1 intégré à LiveNCM, ce même volume est réduit de 55% ce qui ne représente plus que 663 Mo de données échangées. Chaque noeud n'enverra en moyenne que 66 ko contre 146 ko d'échantillons sur une période de 3 mois. Si l'on considère que 75% de l'énergie est consommée par le médium de communication, avec cette méthode, l'autonomie du noeud ne peut qu'augmenter. La Figure 4.3 présente la consommation énergétique du médium de communication après l'utilisation d'un estimateur. On considère que sur 100% de l'énergie d'un noeud : 25% est utile au fonctionnement général (application, processeur, mémoire, capteur...) et 75% est lié aux communications. On remarque alors que l'énergie utile aux transmissions est fortement réduite, de -9% à -86%.

Dans ces conditions, une application simple sans estimateur fonctionnant 365 jours verra son autonomie passer au minimum à 390 jours jusqu'à un maximum de 601 jours (Figure 4.4) dans les conditions d'évaluation citées précédemment. Les estimateurs permettent donc d'agir sur l'autonomie des noeuds en réduisant les transmissions de données. Par exemple, le capteur n°13 qui embarque 2 capteurs a besoin d'échanger 66.5% des échantillons ce qui ne produit qu'un gain énergétique de 7%. Cependant, même dans ce cas de figure, l'autonomie de ce noeud est augmentée de 25 jours. Avec ces résultats, l'utilisateur pourra estimer l'autonomie globale du réseau en regardant celle du noeud le plus faible en fonction d'un précédent relevé de la grandeur à observer.

Pour conclure cette partie, l'ajout d'un estimateur pour permettre de réduire les volumes de données est une piste très intéressante. Avec des gains plus ou moins importants (de 7 à 65%), les

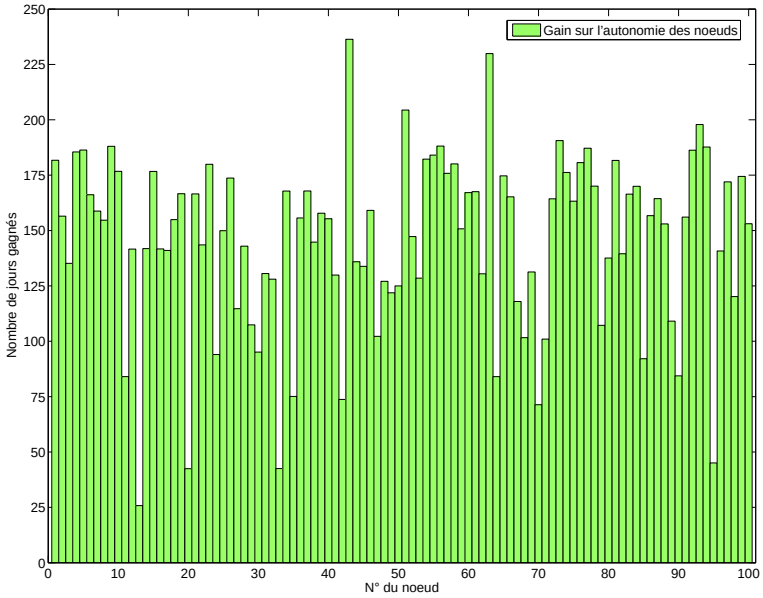

Figure 4.4 – Impact de l'estimation sur l'autonomie d'un noeud

noeuds bénéficient d'une amélioration non négligeable de leur autonomie. Par ailleurs, l'impact sur le volume de données (de -9% à -86%) permet d'améliorer les temps de transfert tout en réduisant le phénomène d'engorgement, c'est-à-dire que l'on réduit les risques d'atteindre la capacité maximale de traitement de la passerelle. Cependant, cette technique ne permet pas d'éliminer totalement ce point, car tous les noeuds peuvent envoyer des données en un même instant et ainsi surcharger la passerelle ou le noeud maître. Elle retarde alors l'engorgement en envoyant moins de données au même instant.

4.5 Projet NeT-ADDeD

Pour appuyer les résultats théoriques, nous avons choisi de déployer la solution d'administration LiveNCM sur une plateforme expérimentale dans le cadre du projet européen FP6 - NeT-ADDeD[3] "New Technologies to Avoid Digital Division in e-Divided areas". Ce projet vise à faciliter l'accès aux informations aux populations et aux zones éloignées de toute infrastructure digitale afin de favoriser le développement qu'il soit humain ou agricole. Le Cemagref ainsi que le LIMOS ont été retenus pour ce projet afin de proposer des solutions technologiques aux échanges dans un milieu rural notamment pour le pilotage de l'irrigation et le suivi à distance des parcelles agricoles. Dans cette optique, les RCSFs sont présentés comme une solution pratique pour les relevés des données nécessaires à répondre aux objectifs précédents. Les LiveNodes ont

[3]http ://www.netadded-project.eu/

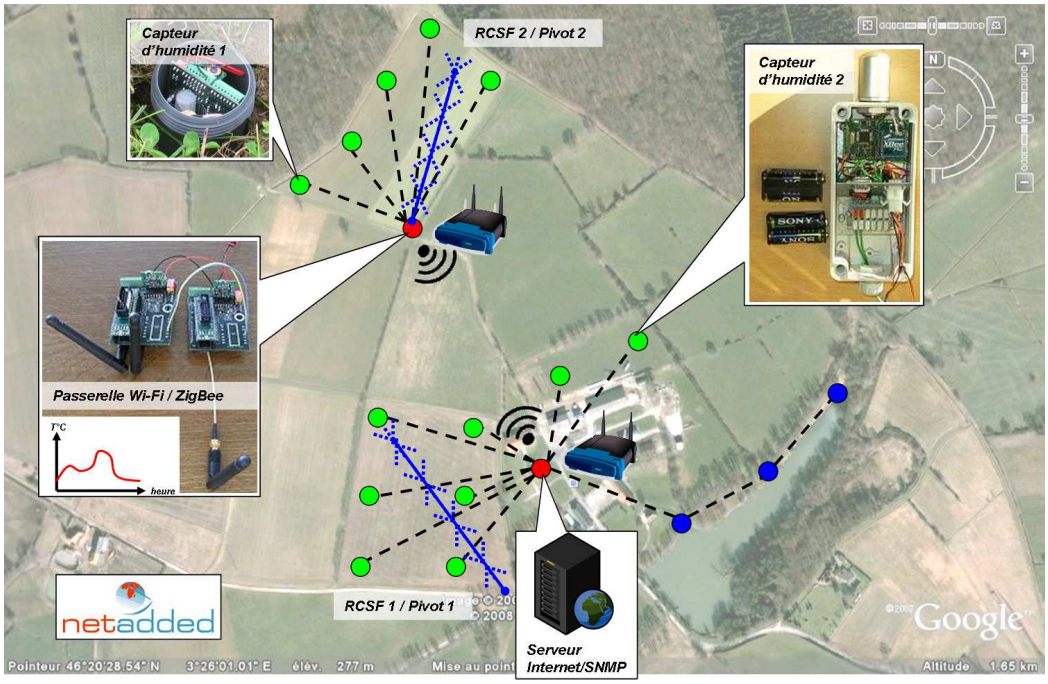

Figure 4.5 – Implantation des capteurs dans le cadre du projet NeT-ADDeD

alors été sélectionnés pour remplir cette tâche et effectuer le relevé de l'humidité du sol d'une parcelle agricole. Afin d'évaluer cette solution, le site de Montoldre du Cemagref a été utilisé pour effectuer les différentes expérimentations. Un schéma de répartition des capteurs présenté par la Figure 4.5 a été retenu pour évaluer les différents points de l'application. Le RCSF est constitué de noeuds relevant l'humidité du sol, la température ou le niveau d'eau, de noeuds relais et d'un serveur central.

Les différents noeuds avec leurs capteurs doivent être enterrés afin de permettre le travail des engins agricoles sans avoir à les déplacer à chaque intervention. Le recouvrement des antennes a donc ajouté une nouvelle source de perturbations ce qui a réduit nécessairement leurs performances en fonction du type de sol. C'est pourquoi, le réseau dispose de noeuds relais au plus proche des parcelles afin de relayer les informations plus efficacement vers le serveur central. Ces plateformes seront en charge de transférer les données acquises par les noeuds capteurs sur de plus longues distances. De plus, les relevés pourront piloter directement, via des vannes électroniques, les solutions d'irrigation disponibles (pivots d'arrosage, compte goutte, canons à eau, etc.) afin d'apporter la quantité nécessaire et suffisante d'eau aux cultures. Par ailleurs, l'autonomie des noeuds est un point crucial de cette application, car ils doivent pouvoir fonctionner plusieurs mois, en général le temps d'une saison de culture, sans intervention extérieure. Il est difficilement envisageable d'imposer à une personne de parcourir souvent les différentes parcelles afin de changer la source d'alimentation.

La solution d'administration LiveNCM se place alors comme l'outil d'administration pouvant améliorer la supervision de réseaux d'OICs. L'estimateur intégré permet d'éliminer les

échantillons inutiles ce qui augmentera l'autonomie du noeud. Par ailleurs, la gestion centralisée via un sous-agent SNMP central permettra un accès aux données rapide et facile pour l'utilisateur sur une machine locale ou à distance. Les échanges entre les noeuds et la passerelle de communication se feront à l'aide d'un noeud évolué utilisant 2 médiums de communication. On se trouve alors dans le 3ème cas de déploiement évoqué par la Figure 3.13. Ce noeud est implanté sur le pivot d'arrosage ce qui permet d'avoir une source d'énergie inépuisable. La liaison avec le sous-agent SNMP s'effectuera donc par un médium Wi-Fi. La solution LiveNCM a été utilisée à la fois pour gérer la vie des noeuds mais aussi pour collecter les données des capteurs en les stockant dans une base de données externe. Les relevés de température et d'humidité sont effectués chaque heure par les passerelles et les noeuds. La solution mise en place depuis 6 mois a permis d'effectuer un relevé de température sur une parcelle agricole. La plateforme LiveNode utilisait un capteur d'humidité résistif sur lequel nous n'avons pu avoir de relevé valide sur une longue période à cause d'une chaîne de mesure défaillante. Les graphes de la Figure 4.6 donnent un relevé de température d'un pivot sur une durée d'un mois. Le graphe du haut superpose les mesures relevées avec les mesures estimées par l'estimateur de LiveNCM. Le graphe du bas affiche les échantillons qui seront effectivement envoyés durant cette période. La marge d'erreur a été fixée à ± 0.5°C soit la moitié de la résolution du capteur. La campagne de mesure comporte 4465 échantillons dont seulement 506 sont utiles pour reconstruire la courbe. La période de synchronisation est alors fixée à 24h.

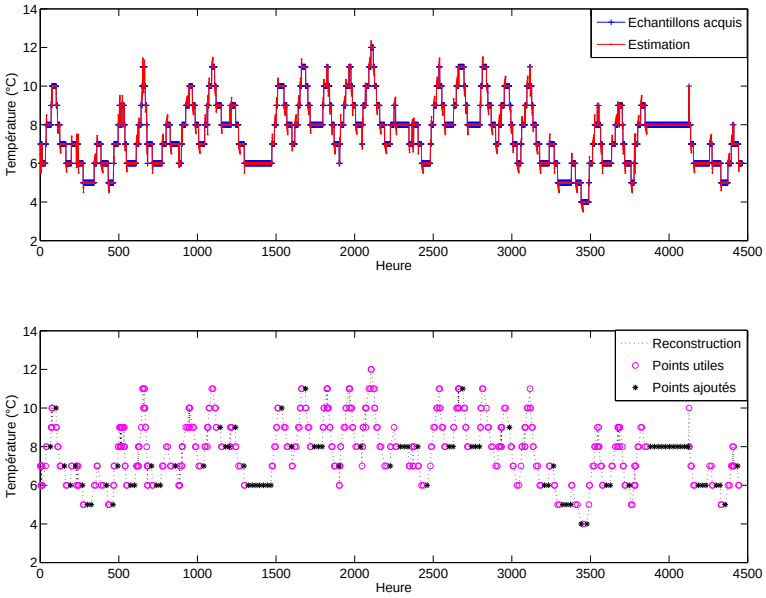

Figure 4.6 – Relevé de température sur un mois : haut, superposition des valeurs relevées avec les valeurs estimées ; bas, valeurs envoyées

Afin de mettre à disposition les différentes données administrées, nous avons produit une

interface de communication simple. Cette interface fera appel aux différentes fonctionnalités SNMP pour accéder aux données sauvegardées par l'agent. Pour permettre une utilisation simple et facile à distance, nous avons choisi de la développer sur un serveur Internet du type Apache. Le langage utilisé est PHP : Hypertext Preprocessor (PHP) pour la gestion des pages couplé au langage Asynchronous JavaScript and XML (AJAX) pour fluidifier la navigation de l'utilisateur. Par ailleurs, elle est complétée par sa propre base de données pour réaliser des historiques des mesures. En effet, le protocole SNMP ne représente l'image d'un système qu'à l'instant t, n'ayant pas de capacités de sauvegarde, les données sont donc écrasées à l'instant t+1. Périodiquement, une application passerelle alimente la base de données en fonction des informations présentes dans la MIB du sous-agent SNMP. A période fixe, elle récolte les données capteurs dans la MIB et les insère dans la base de données. Enfin, pour visualiser les positions des différents capteurs, les outils proposés par Google Maps ont été intégrés. L'agent SNMP stocke les postions GPS fournies directement par le noeud s'il dispose d'un module de géo-localisation, sinon une position par défaut est définie lors de la première configuration. L'interface ainsi développée est présentée par la Figure 4.7. Sur la gauche, la liste des différents éléments supervisés est affichée sur laquelle il est possible d'appliquer des filtres de sélection. Ensuite, sur la partie droite, les informations de l'élément sélectionné sont affichées (ici, les informations du capteur 6). L'utilisateur a la possibilité de modifier la fréquence d'échantillonnage pour les capteurs, ainsi que de nombreuses informations sur le médium de communication quand on sélectionne un noeud. Pour les noeuds et les RCSFs, la carte d'identité est accessible pour modification. Évidement, une fonction de suppression est intégrée pour les différents éléments. A la suppression, un message est envoyé au noeud pour valider la suppression et ainsi désactiver toutes les fonctionnalités liées à l'application.

L'expérimentation sur ce projet a permis d'utiliser la solution d'administration LiveNCM sur une application réelle sur plusieurs mois. Sur cette application, nous n'avons constaté aucune perte de message entre les noeuds et les passerelles. Le contrôle d'erreur malgré quelques lacunes de détection n'a causé aucune erreur de traitement. La fiabilité du protocole LiveNCM-P a donc pu être éprouvée ce qui valide les différents choix techniques que ce soit pour la passerelle ou sur l'agent embarqué sur les noeuds. Cependant, le module de mise à jour n'a pas pu être évalué sur cette application. En définitive, les parties communication et traitement des messages ont été validées. Avec des premiers résultats concluants, on peut envisager d'exploiter la solution d'administration LiveNCM sur différents RCSFs pour différentes applications. De plus, l'interface développée permet d'accéder à distance aux informations des noeuds afin de suivre l'évolution du réseau sans avoir à se déplacer sur le terrain.

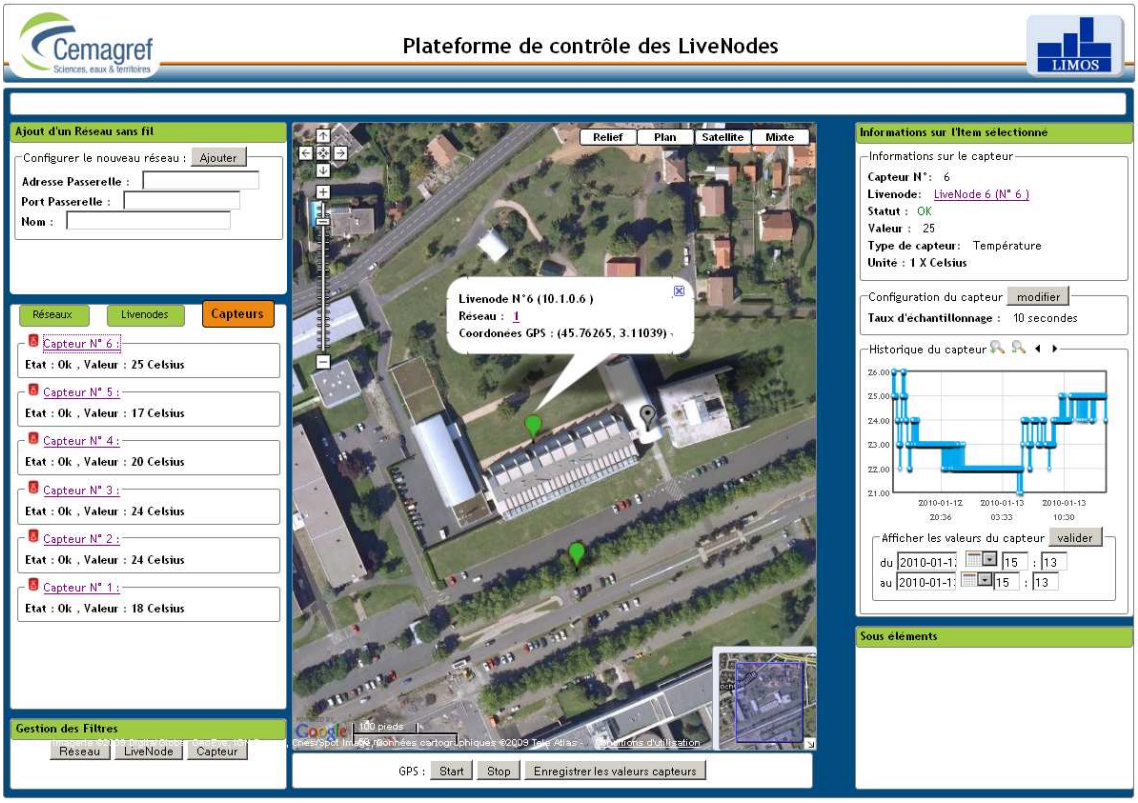

Figure 4.7 – Interface d'administration des LiveNodes

4.6 Conclusion

Dans ce chapitre, nous avons évalué les différentes parties de la solution d'administration LiveNCM. L'architecture retenue, avec une partie SNMP, centralisant les données des noeuds à laquelle est associée une passerelle de communication, offre des performances intéressantes. De plus, le protocole de communication développé LiveNCM-P propose des solutions innovantes pour améliorer l'autonomie des noeuds. En ne transmettant que les échantillons utiles en fonction d'une marge d'erreurs fixée par l'utilisateur, les noeuds voient leur autonomie s'améliorer. Sur un réseau vaste simulant quelques milliers de noeuds, le gain énergétique évolue de 7% jusqu'à 93%. Le choix d'un estimateur pour cette tâche est alors pertinent puisqu'en plus de réduire les volumes échangés, il permet d'améliorer l'autonomie du noeud et de réduire les risques de saturations du réseau. La deuxième technique implémentée par ce protocole s'appuie sur un algorithme de compression reconnu (LZW). Cet algorithme obtient de bonnes performances sur la diffusion des mises à jour. Malgré un temps de compression long, il permet de réduire de près de 50% (de 40 à 50% sur les différents fichiers compressés) la taille du fichier ce qui réduit aussi le temps nécessaire à l'envoi. La décompression, par contre, est très rapide ce qui permet une utilisation réduite des ressources du noeud pour reconstruire l'application.

La solution d'administration LiveNCM a pu être évaluée à la fois en simulation et en pratique ce qui a permis de vérifier les différents résultats. Les limitations de cette solution sont souvent liées aux techniques employées pour relever la grandeur ou pour traiter les messages échangés. L'échantillonnage des signaux observés est le point le plus important qui va influencer les performances de l'estimateur embarqué. Les échanges de données dans le réseau sont sujets à différentes perturbations qui seront prises en compte par le protocole LiveNCM-P via un CRC classique. Un noeud traitera un message intègre et bien formé. La détection des messages s'appuyant sur plusieurs caractères permet d'éliminer les faux messages que l'on pourrait détecter avec seulement un caractère de début de trame. Enfin, le module de mise à jour déployé sur les noeuds par l'intermédiaire de leur agent, obtient des performances théoriques intéressantes qui devront être confirmées en conditions réelles.

Au final, LiveNCM propose des choix technologiques différents des solutions existantes comme ANMP ou GUERRILLA. La centralisation des données sur un seul et même agent SNMP permet une meilleure réactivité aux requêtes de l'utilisateur. De cette façon, la charge de calcul et de mémoire sur chacun des noeuds est réduite ce qui permet d'adopter des traitements plus complexes et des applications plus évoluées. Par ailleurs, en choisissant de remplacer les messages lourds de SNMP par des messages mieux adaptés aux contraintes des RCSFs, on réduit les volumes échangés ainsi que les probabilités d'erreurs dans les transmissions. Le second intérêt est de diminuer le nombre de collisions et la charge réseau. Les noeuds sont alors identiques par leur plateforme et par leur application. Sur chaque noeud, un agent léger d'administration est déployé pour remplacer l'agent SNMP afin de répondre aux différentes requêtes de données ou d'administration.

$\sim \star \sim \star \sim \star \sim$

Conclusion et perspectives

Le bond technologique de ces dernières années a permis le développement de nombreux outils matériels et logiciels innovants pour récolter les grandeurs physiques dans divers milieux (contrôlés mobiles, hostiles, difficiles, etc.) afin d'améliorer l'utilisation des ressources naturelles, ou des infrastructures routières par exemple. C'est ainsi, que les RCSF ont vu le jour et constituent aujourd'hui un domaine de recherche en pleine expansion. La variété des applications entraîne de contraintes multiples, influençant l'évolution des grandeurs observées mais aussi des plateformes matérielles utilisées.

Pour répondre à ces différentes contraintes, de nombreuses solutions logicielles ont été développées pour améliorer les communications, la gestion des données et l'organisation des réseaux. Cependant, l'aspect supervision a été moins exploré au profit des méthodes internes aux RCSF comme les protocoles de routage, ou d'auto-configuration. Aujourd'hui, leur maturité oblige de réaborder cet aspect afin de pouvoir déployer des réseaux viables. De plus, avec une densité de noeuds sur un même site pouvant varier de quelques dizaines à plusieurs milliers, la charge de maintenance est importante et doit donc être effectuée de manière simple, efficace et à distance. C'est pourquoi, cette thèse s'est focalisée sur la supervision de réseaux d'OICs ainsi que sur les différentes méthodes permettant une meilleure utilisation de la ressource énergétique de chaque noeud. Menée en collaboration avec l'unité de recherche TSCF du Cemagref de Clermont Ferrand et le LIMOS UMR 6158 CNRS, elle a eu pour objectif d'étudier une solution d'administration pour les RCSF afin de permettre un déploiement à large échelle tout en proposant un accès simple et peu intrusif aux différents noeuds. La généricité de cette solution a été un point important pour permettre son utilisation dans divers contextes applicatifs et environnementaux.

Bilan

La solution d'administration proposée, LiveNCM, a pour objectif principal de faciliter l'accès aux données sensibles des noeuds par un simple utilisateur en limitant l'aspect intrusif des méthodes d'administration existantes. Les opérations de maintenance pouvant devenir très rapidement lourdes, il a été nécessaire de voir comment les rendre transparentes et légères. Pour les RCSF, les solutions existantes basées sur un protocole existant ou sur une application spécifique ne répondent plus à toutes les contraintes. Les premières sont trop lourdes ce qui réduit grandement l'application embarquée et les secondes sont liées à une plateforme matérielle ou à

un système d'exploitation spécifique. La portabilité de ces solutions est alors faible. Par ailleurs, les systèmes de données à la demande inclus dans ces méthodes influencent le fonctionnement des noeuds en augmentant les temps de traitements liés aux requêtes utilisateurs.

Nous avons donc présenté une nouvelle réponse à la problématique d'administration dans les RCSF, LiveNCM. Basée sur le protocole SNMP pour sa partie centrale et son interface 'utilisateur', elle emploie ensuite un protocole de communication léger et adapté. En utilisant un codage des informations moins coûteux, les messages sont diffusés plus rapidement car moins long et donc usent moins d'énergie. L'une des caractéristiques de LiveNCM consiste à employer des mécanismes simples pour réduire les données échangées au sein du réseau ainsi que celles qui seront sauvegardées sur chaque noeud. Le premier appelé diagnostic indirect, considère l'ensemble des messages comme source d'information d'administration. Pour chaque échange, ce mécanisme peut déduire des informations d'état par exemple du capteur ou du noeud. Pour chaque information récoltée de cette façon, une requête d'administration sera éliminée. Naturellement, un paramétrage permet de définir l'importance des données et le noeud enverra, en fonction de ces paramètres, les données ayant changées. En complément, les estimateurs constituent un moyen innovant et efficace de contrôle des informations utiles. Avec des performances importantes, cette technique permet de préserver l'énergie des noeuds et de réduire les risques d'engorgement du réseau. Basée sur une fonction simple, le coût de la mise à jour du modèle d'estimation est négligeable face au coût d'envoi d'un message. Par la suite, dans certains cas, les estimateurs ne peuvent pas être utilisés ce qui nous oblige à nous appuyer sur un algorithme de compression simple et léger afin de limiter son impact sur les performances du noeud. Enfin, la liaison entre les noeuds et l'agent SNMP passe par l'intégration au réseau d'une passerelle de communication. La conception de cet élément a été effectuée de manière à pouvoir dans un premier temps, transférer les messages et, dans un second temps, soutenir des fonctionnalités d'administration. Élaborée à partir de briques logicielles réutilisables, nous avons proposé une technique de développement originale visant à utiliser un maximum de codes communs entre les parties fixes et embarquées de LiveNCM. De cette façon, le développeur pourra déployer aussi facilement une passerelle sur une machine fixe que sur un noeud évolué.

Testée et validée sur une application utilisant un réseau simple de quelques noeuds, la solution d'administration LiveNCM a pu prouver son efficacité et sa stabilité en milieu ouvert non contrôlé. La plateforme d'évaluation utilisée a été le LiveNode sans système d'exploitation ni protocole de routage. En procédant ainsi, on élimine les risques de lier ces différents éléments afin de pouvoir déployer notre solution d'administration dans des contextes différents. Malgré de nombreux obstacles (arbres, terre, structures métalliques, etc.) et des conditions changeantes, les performances de LiveNCM sont importantes d'un point de vue communication avec une diminution significative du trafic dans le réseau et un faible taux de pertes de messages.

Perspectives

La solution d'administration LiveNCM dispose d'une base de développement solide constituant une nouvelle méthode de supervision de réseaux d'OICs. L'orientation choisie pour cette thèse a été la gestion énergétique des noeuds pour réduire au maximum l'impact de l'admi-

nistration sur le RCSF et contrôler son autonomie. Nous choisissons alors d'effectuer plus de calculs pour envoyer moins de données afin de réduire les coûts de la communication sans fil. Cependant, certains points restent encore à être approfondis pour optimiser cet aspect. Le mécanisme de diagnostic indirect inclus dans LiveNCM demande d'étudier en profondeur les liens entre les différentes informations des noeuds afin de maximiser le nombre de données déduites des différents messages tout en minimisant la taille de ces derniers. Par ailleurs, les politiques énergétiques décrites dans ce mémoire restent encore au stade expérimental et doivent être plus largement validées. L'étape suivante sera donc de les mettre en place tout en augmentant les points d'action pour réduire la consommation du noeud. Si pour le moment, elles se limitent à faire varier les fréquences d'envoi et de fonctionnement, il est évident que d'autres points, comme les puissances d'émission ou la gestion plus fine des périphériques utiles, doivent être étudiés. De plus, les données administrées actuellement par le prototype sont non exhaustives. Une plus large étude permettra de mettre en évidence des paramètres complémentaires à collecter, afin d'affiner l'état des noeuds à chaque instant. Par ailleurs, lors de la conception du LiveNode, le coût monétaire de la plateforme a influencé les choix d'architecture et de composants. Une plateforme mieux optimisée ou totalement intégrée permettra alors d'augmenter significativement les bénéfices sur l'autonomie et les communications de notre solution.

Plus globalement, l'expansion des réseaux LAN et WAN pose des problèmes d'adressage des noeuds. Peu à peu, les RCSF apparaîtront sur nos réseaux comme des machines où chaque noeud disposera d'une adresse unique. L'adressage actuel de type IPv4 arrive à saturation ce qui interdisait un accès distant à chaque élément du réseau. Pour y remédier, IPv6 vient prendre le relais en fournissant beaucoup plus d'adresses, rendant possible l'apparition des RCSF sur Internet ou sur des réseaux locaux. Une prochaine évolution de LiveNCM sera donc sa migration vers IPv6 pour permettre un accès distant à une passerelle ou à un agent SNMP. La pile de protocoles 6LoWPAN sera sans doute le pilier de cette migration. Les premières pistes de travaux concernant l'adaptation du protocole SNMP à celui-ci ainsi que l'apparition de plateformes matérielles réalisant son intégration,tendent à confirmer ceci. Le futur des RCSF devrait passer par cette migration.

Un dernier point sans réponse actuellement est la sécurisation des échanges dans ces réseaux. Dans le cadre de cette thèse, le protocole de communication n'intègre aucune information de sécurisation. La sécurité constitue une problématique entière qui devra être intégrée, au fil du temps, dans les échanges entre les noeuds mais aussi avec le superviseur. Pour répondre dans un premier temps à ce sujet, le protocole SNMP utilisé en version v2c devra migrer vers la version v3 pour inclure une gestion fine des droits d'accès et un cryptage des trames échangées. En aval, la sécurisation passera par la définition de techniques de cryptage simples, efficaces et surtout peu intrusives. Ce point plus sensible disposera alors de méthodes plus spécifiques de contrôle d'intégrité de message dispensant ainsi le protocole de communication LiveNCM-P de cette tâche.

Enfin, dans un futur proche, LiveNCM sera intégré dans une solution complète de RCSF avec un système d'exploitation événementiel et multitâche comprenant un protocole de routage adapté ainsi qu'un algorithme d'auto-configuration. La plateforme matérielle LiveNode

devrait disposer d'une architecture multi-processeur plus efficace énergétiquement et mieux armée contre les pannes et autres erreurs matérielles ou logicielles. Les applications visées seront alors plus complexes avec un fort potentiel de décision sur l'environnement comme le pilotage d'engins agricoles ou la gestion de l'irrigation. Enfin, pour compléter les expérimentations, une nouvelle application sur le suivi hydrique d'un vignoble permettra d'éprouver l'ensemble des mécanismes de LiveNCM dans un milieu agricole.

$\sim \star \sim \star \sim \star \sim$

Bibliographie

[Akyildiz 02] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam & E. Cayirci. *Wireless sensor networks : a survey.* In Comput. Networks, volume 38, pages 393–422, 2002.

[Akyildiz 06] Ian F. Akyildiz & Erich P. Stuntebeck. *Wireless underground sensor networks : Research challenges.* In Ad Hoc Networks, volume 4, pages 669–686, 2006.

[Al-Karaki 04] J. N. Al-Karaki & A. E. Kamal. *Routing techniques in wireless sensor networks : a survey.* In Wireless Communications, IEEE, volume 11, pages 6–28, 2004.

[ATMEL 05] ATMEL. *ATMEL AT91SAM7S256*, nov.2006 2005.

[ATMEL 09] ATMEL. *ATmega168 Automotive*, 2009.

[Barr 02] R. Barr, J. C. Bicket, D. S. Dantas, B. Du, T. W. D. Kim, B. Zhou & E. Gün Sirer. *On the need for system-level support for ad hoc and sensor networks.* In SIGOPS Oper. Syst. Rev., volume 36, pages 1–5, 2002.

[Beckwith 04] R Beckwith. *Report from the Field : Results from an Agricultural Wireless Sensor Network.* In Teibel Dan & Bowen Pat, editeurs, 29th Annual IEEE International Conference on Local Computer Networks (LCN'04), pages 471–478, Tampa, Florida, U.S.A., 2004.

[Blumenthal 02] U. Blumenthal & B. Wijnen. *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3).* volume RFC 3414, pages 1–88, 2002.

[Bokareva 06] T. Bokareva, W. Hu, S. Kanhere, B. Ristic, T. Bessell, M. Rutten & S. Jha. *Wireless sensor networks for battlefield surveillance.* In in Proc. of the Land Warfare Conference, 8 pages, Brisbane, Australia, 2006.

[Boulis 03] A. Boulis & M.B. Srivastava. *Node-Level Energy Management for Sensor Networks in the Presence of Multiple Applications.* In IEEE PerCom 2003, pages 41 – 49, Fort Worth, Texas, USA, 2003.

[Brunet 01] J. Brunet, L. Talazac, V. Battut, A. Pauly, J. P. Blanc, J. P. Germain, S. Pellie & Soulier C. *Evaluation of atmospheric pollution by two semiconductor gas sensors.* In Thin Solid Films, volume 391, pages 308–313. Elsevier, 2001.

[Case 90] J. Case, M. Fedor, M. Schoffstall & J. Davin. *A Simple Network Management Protocol.* volume RFC 1157, pages 1–36, 1990.

[Case 93] J. Case, K. McCloghrie, M. Rose & S. Waldbusser. *Introduction to version 2 of the Internet-standard Network Management Framework.* volume RFC 1441, pages 1–13, 1993.

[Chaczko 05] Z. Chaczko & F. Ahmad. *Wireless Sensor Network Based System for Fire Endangered Areas.* In the Third International Conference on Information Technology and Applications (ICITA'05), pages 203–207, Sydney, Australia, 2005.

[Chanet 07] J. P. Chanet. *Algorithme de routage coopératif à qualité de service pour des réseaux ad hoc agri-environnementaux.* Doctorat informatique, université blaise pascal, clermont ii, 2007.

[Chelius 09] G. Chelius, E. Fleury, A. Fraboulet & J. C. Lucet. *A wireless sensor network to measure the health care workers exposure to tuberculosis.* In International Workshop and Conference on Complex Networks and their Applications (NetSci 09), Abstract pages, Venise, Italy, 2009.

[Chen 99] W. Chen, N. Jain & S. Singh. *ANMP : Ad Hoc Network Management Protocol.* In Selected Areas in Communications, IEEE Journal on, 26 pages, 1999.

[Chien-Liang 05] F. Chien-Liang, R. Gruia-Catalin & L. Chenyang. *Mobile agent middleware for sensor networks : an application case study.* In Proceedings of the 4th international symposium on Information processing in sensor networks, 6 pages, Los Angeles, California, 2005. IEEE Press.

[Chong 03] C. Chong & S. Kumar. *Sensor Networks : Evolution, Opportunities, and Challenges.* In Proceedings of the IEEE, volume 91, pages 1247–1256, 2003.

[Clausen 03] T. Clausen & P. Jacquet. *Optimized Link State Routing Protocol (OLSR).* volume RFC 3626, pages 1–75, 2003.

[Coleri 04] S. Coleri, S. Y. Cheung & P. Varaiya. *Sensor Networks for Monitoring Traffic.* In In Forty-Second Annual Allerton Conference on Communication, Control, and Computing, pages 1–10, Univ. of Illinois, 2004.

[Crossbow 09] Crossbow. *MICAz - http ://www.xbow.com/*, 2009.

[De Sousa 07] G. De Sousa, H. Zhou, K. M. Hou, C. De Vaulx & J. P. Chanet. *Adaptive System for wireless Sensor Networks Applications.* In Journal of Harbin Institute of Technology, volume 39, pages 154–157, 2007.

[De Sousa 08] G. De Sousa. *Etude en vue de la réalisation de logiciels bas niveau dédiés aux réseaux de capteurs sans fil : microsystème de fichiers.* Doctorat en informatique, université blaise pascal clermont ii, 2008.

[Deb 01] B. Deb, S. Bhatnagar & B. Nath. *A Topology Discovery Algorithm for Sensor Networks with Applications to Network Management.* Dcs technical report dcs-tr-441, Department of Computer Science, Rutgers University, 2001.

[Diao 07] X. Diao, E. Lai, K. M. Hou & H. Zhou. *An Auto-Clustering Algorithm For Wireless Sensor Network Management Protocol.* In 7ème Conférence Internationale sur les NOuvelles Technologies de la Repartition (NOTERE'07), Marrakech, MAR, 4-8 juin, 2007, 6 pages, 2007.

[Diao 08] X. Diao, J. J. Li, K. M. Hou, H. Zhou & A. Jacquot. *Cooperative Inter-Vehicle Communication Protocol Dedicated to Intelligent Transport Systems.* In New Technologies, Mobility and Security, 2008. NTMS '08., pages 1–5, 2008.

[Dubois-Ferriere 06] H. Dubois-Ferriere, R. Meier, L. Fabre & P. Metrailler. *TinyNode : A Comprehensive Platform for Wireless Sensor Network Applications.* In Proceedings of the fifth international conference on Information processing in sensor networks IPSN 2006, pages 358–365, Nashville, Tennessee, USA, 2006. ACM Press.

[Ellis 02] J. Ellis, K. Siwiak & R. Roberts. *IEEE 802.15 WPAN High Rate Alternative PHY Task Group 3a (TG3a)*, 2002.

[Erdogan 03] A. Erdogan, E. Cayirci & V. Coskun. *Sectoral sweepers for sensor node management and location estimation in adhoc sensor networks.* In Military Communications Conference, 2003. MILCOM 2003. IEEE, volume 1, pages 555–560, Turkish Naval Acad., Istanbul, Turkey, 2003.

[Eriksson 08] L. Eriksson, M. Elmusrati & M. Pohjola. *Introduction to wireless automation - Collected papers of the spring 2007 postgraduate seminar.* Rapport technique Report 155, Helsinki University of Technology, Department of Automation and Systems Technology, 2008.

[Feeney 01] L. M. Feeney & M. Nilsson. *Investigating the energy consumption of a wireless network interface in an ad hoc networking environment.* In INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, volume 3, pages 1548–1557 vol.3, 2001.

[García-Hernando 08] Ana-Belén García-Hernando, José-Fernán Martínez-Ortega, Juan-Manuel López-Navarro, Aggeliki Prayati & Luis Redondo-López. Problem solving for wireless sensor networks. Computer Communications and Networks. Springer, 2008.

[Hadim 06] S. Hadim & N. Mohamed. *Middleware : middleware challenges and approaches for wireless sensor networks.* Distributed Systems Online, IEEE, vol. 7, no. 3, 2006.

[Heinzelman 00] W. R. Heinzelman, A. Chandrakasan & H. Balakrishnan. *Energy-efficient communication protocol for wireless microsensor networks.* In System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on, volume 2, pages 1–10, 2000.

[Hill 01] J. Hill & D. Culler. *A wireless embedded sensor architecture for system-level optimization.* In Technical report, pages 1–12, U.C. Berkeley, 2001.

[Hong 03] S. Hong & T.-H. Kim. *SenOS : State-driven Operating System Architecture for Dynamic Sensor Node Reconfigurability.* In International Conference on Ubiquitous Computing (ICUC)., pages 201 – 203, Seoul, Korea, 2003.

[Hou 07a] K. M. Hou, G. De Sousa, J. P. Chanet, H. Zhou, M. Kara, A. Amamra, X. Diao, C. De Vaulx, J. J. Li & A. Jacquot. *LiveNode : LIMOS versatile embedded wireless sensor node.* Journal of Harbin Institute of Technology, vol. 39, 2007.

[Hou 07b] K. M. Hou, G. De Sousa, J. P. Chanet, H. Zhou, M. Kara, A. Amamra, X. Diao, C. De Vaulx, J. J. Li & A. Jacquot. *LiveNode : LIMOS versatile embedded wireless sensor node.* In Workshop International sur Les Réseaux de Capteurs sans Fil en conjonction avec la 7ème Conférence Internationale sur les NOuvelles TEchnologies de la REpartition (NOTERE), Marrakech, MAR, 4 juin 2007, 5 pages, 2007.

[Huffman 52] D. A. Huffman. *A Method for the Construction of Minimum-Redundancy Codes.* In Proceedings of the IRE, volume 40, pages 1098–1101, 1952.

[IEA 09] IEA. *International Energy Agency : Photovoltaic Power Systems Programme Task 2, Performance, Reliability and Analysis of Photovoltaic Systems - http ://www.iea-pvps.org/*, 2009.

[IEEE 03] IEEE. *IEEE Standard for Information Technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific equirements-Part 15.4 : Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs).* pages 1–679, 2003.

[IEEE 05] IEEE. *IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements Part 15.1 : Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPAN s).* pages 1–600, 2005.

[IEEE 07] IEEE. *IEEE 802.11-2007, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, June 2007.* pages 1–1232, 2007.

[IEEE 09] IEEE. *IEEE Standard for Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5 : Enhancements for Higher Throughput.* In IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009), pages 1–502, 2009.

[Jacquot 07a] A. Jacquot, J. P. Chanet, K. M. Hou, X. Diao & D. Boffety. *Issue of wireless sensor network management in agri-environmental applications.* In EFITA 2007, Glasgow, Scotland, GBR, 2-5 July 2007, 2 pages, 2007.

[Jacquot 07b] A. Jacquot, J. P. Chanet, K. M. Hou & H. Zhou. *Un objet communicant intelligent pour des réseaux agri-environnementaux : LiveNode.* In 5ème édition des journées STIC & Environnement, Lyon, 13-15 novembre 2007, 9 pages, 2007.

[Jacquot 08] A. Jacquot, J. P. Chanet, K. M. Hou, X. Diao & J. J. Li. *A new approach for Wireless Sensor Network management : LiveNCM.* In Second IFIP international conference on New Technologies, Mobility and Security NTMS'08, Wireless Sensor Network workshop, 6 pages, Tangier, Morocco, 2008.

[Jacquot 09] A. Jacquot, J. P. Chanet, K. M. Hou, Diao Xunxing & Li Jian-Jin. *LiveNCM : A new wireless management tool.* In AFRICON, 2009. IEEE AFRICON '09., pages 1–6, Nairobi, Kenya, 2009.

[Jiang 04] Q. Jiang & D. Manivannan. *Routing protocols for sensor networks.* In Consumer Communications and Networking Conference, 2004. CCNC 2004. First IEEE, pages 93–98, 2004.

[Jin-Shyan 07] L. Jin-Shyan, S. Yu-Wei & S. Chung-Chou. *A Comparative Study of Wireless Protocols : Bluetooth, UWB, ZigBee, and Wi-Fi.* In Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE, pages 46–51, 2007.

[Kahn 99] J. M. Kahn, R. H. Katz & K. S. J. Pister. *Next century challenges : mobile networking for "Smart Dust".* In Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking, pages 271–278, Seattle, Washington, United States, 1999. ACM.

[Kim 07a] K. Kim, G. Montenegro, S. Park, I. Chakeres & C. E. Perkins. *Dynamic MANET On-Demand for 6LoWPAN (DYMO-low) Routing.* In draft-montenegro-6lowpan-dymo-low-routing-03, pages 1–17, 2007.

[Kim 07b] K. Kim, S. Park, G. Montenegro, S. Yoo & N. Kushalnagar. *6LoWPAN Ad Hoc On-Demand Distance Vector Routing (LOAD).* In draft-daniel-6lowpan-load-adhoc-routing-03, pages 1–17, 2007.

[Kim 07c] K. Kim, S. Yoo, S. Park, J. Lee & G. Mulligan. *Hierarchical Routing over 6LoWPAN (HiLow).* In draft-daniel-6lowpan-hilow-hierarchical-routing-01, pages 1–12, 2007.

[Kulik 02] J. Kulik, W. Heinzelman & H. Balakrishnan. *Negotiation-based protocols for disseminating information in wireless sensor networks.* In Wirel. Netw., volume 8, pages 169–185, 2002.

[Lee 06] W. L. Lee, A. Datta & R. Cardell-Oliver. *Network Management in Wireless Sensor Networks.* In Denko M. K. & Yang L. T., editeurs, Handbook of Mobile Ad Hoc and Pervasive Communications, pages 1–20. American Scientific Publishers, USA, 2006.

[Levis 02] P. Levis & D. Culler. *MATE : a tiny virtual machine for sensor networks.* In SIGOPS Oper. Syst. Rev., volume 36, pages 85–95, 2002.

[Levis 05] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer & D. Culler. *TinyOS :An Operating System for Sensor Networks.* In Ambient Intelligence, pages 115–148. Springer Berlin Heidelberg, 2005.

[Liu 03] T. Liu & M. Martonosi. *Impala : a middleware system for managing autonomic, parallel sensor systems.* In Proceedings of the ninth ACM SIGPLAN symposium on Principles and practice of parallel programming, pages 107–118, San Diego, California, USA, 2003. ACM.

[López Riquelme 09] J. A. López Riquelme, F. Soto, J. Suardíaz, P. Sánchez, A. Iborra & J. A. Vera. *Wireless Sensor Networks for precision horticulture in Southern Spain.* In Computers and Electronics in Agriculture, volume 68, pages 25–35, 2009.

[Madden 05] S. Madden, M. J. Franklin, J. M. Hellerstein & W. Hong. *TinyDB : an acquisitional query processing system for sensor networks.* In ACM Trans. Database Syst., volume 30, pages 122–173, 2005.

[Mainwaring 02] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk & J. Anderson. *Wireless sensor networks for habitat monitoring.* In Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, pages 88–97, Atlanta, Georgia, USA, 2002. ACM.

[Malan 04] David Malan, Thaddeus Fulford-Jones, Matt Welsh & Steve Moulton. *CodeBlue : An Ad Hoc Sensor Network Infrastructure for Emergency Medical Care.* In International Workshop on Wearable and Implantable Body Sensor Networks, pages 1–4, 2004.

[Matese 09] A. Matese, S. F. Di Gennaro, A. Zaldei, L. Genesio & F. P. Vaccari. *A wireless sensor network for precision viticulture : The NAV system.* In Computers and Electronics in Agriculture, volume 69, pages 51–58, 2009.

[McCloghrie 91] K. McCloghrie & M. Rose. *Management Information Base for Network Management of TCP/IP-based internets : MIB-II.* volume RFC 1213, pages 1–70, 1991.

[Montenegro 07] G. Montenegro, N. Kushalnagar, J. Hui & D. Culler. *Transmission of IPv6 Packets over IEEE 802.15.4 Networks.* volume RFC4944, pages 1–30, 2007.

[Morais 08] R Morais, M. A. Fernandes, S. G. Matos, C. Serôdio, P. J. S. G. Ferreira & M. J. C. S. Reis. *A ZigBee multi-powered wireless acquisition device for remote sensing applications in precision viticulture.* In Computers and Electronics in Agriculture, volume 62, pages 94–106, 2008.

[Net-Added 09] Net-Added. *European Net-Added project.* In FP6 project, http ://www.netadded-project.eu/, 2009.

[Perkins 03] C. E. Perkins, E. Belding-Royer & S. Das. *Ad hoc On-Demand Distance Vector (AODV) Routing.* volume RFC 3561, pages 1–37, 2003.

[Puccinelli 05] D. Puccinelli & M. Haenggi. *Wireless sensor networks : applications and challenges of ubiquitous sensing.* In Circuits and Systems Magazine, IEEE, volume 5, pages 19–31, 2005.

[Ramanathan 05] M. Chhabra J. Kushalnagar N. Krishnamurthy L. Estrin D. Ramanathan N. Yarvis. *A Stream-Oriented Power Management Protocol for Low Duty Cycle Sensor Network Applications.* In Embedded Networked Sensors, 2005. EmNetS-II. The Second IEEE Workshop on, pages 53 – 62, 2005.

[Rivest 92] R. Rivest. *The MD5 Message-Digest Algorithm.* pages 1–21, 1992.

[Rooks 07] J. W. Rooks & R. Linderman. *High Performance Space Computing.* In Aerospace Conference, 2007 IEEE, pages 1–9, 2007.

[Roundy 04] Shad Roundy, Dan Steingart, Luc Frechette, Paul Wright & Jan Rabaey. *Power Sources for Wireless Sensor Networks.* In Springer Berlin/Heidelberg, editeur, Wireless Sensor Networks, volume 2920/2004 of *Lecture Notes in Computer Science*, pages 1–17. 2004.

[Sentillia 09] Sentillia & Moteiv. *TmoteSky - http ://sentilla.com/files/pdf/eol/tmote-sky-datasheet.pdf*, 2009.

[Sethi 01a] A.S. Sethi, D. Zhu & V. Hnatyshin. *Hierarchical Management of Battlefield Networks with the SHAMAN Management System.* In A.J. Tardif & J.W. Gowens, editeurs, Final Report (1996-2001) Advanced Telecommunications and Information Distribution Research Program (ATIRP), pages 7.1–7.28, US Army Research Laboratory, 2001.

[Sethi 01b] A.S. Sethi, D. Zhu & P. Kalyanasundaram. *SHAMAN - An Environment for Distributed Management Applications.* In Integrated Network Management Proceedings, pages 321–324, Seattle, USA, 2001.

[Shen 02] C.C. Shen, C. Jaikaeo, C. Srisathapornphat & H. Huang. *The Guerrilla Management Architecture for Ad hoc Networks.* In MILCOM 2002, volume 1, pages 467–472, 2002.

[Song 05] H. Song, D. Kim, K. Lee & J. Sung. *UPnP-Based Sensor Network Management Architecture.* In Second International Conference on Mobile Computing and Ubiquitous Networking, 6 pages. ICMU, 2005.

[Stojmenovic 05] I. Stojmenovic. Handbook of sensor networks : Algorithms and architectures. Wiles Series on Parallel and Distributed Computing. 2005.

[Sun 09] Y. Sun, L. Li, P. Schulze Lammers, Q. Zeng, J. Lin & H. Schumann. *A solar-powered wireless cell for dynamically monitoring soil water content.* In Computers and Electronics in Agriculture, volume 69, pages 19–23, 2009.

[Texas-Instruments 09] Texas-Instruments. *MSP430 16-bit Ultra-Low Power MCUs*, 2009.

[Tolle 05] G. Tolle & D. Culler. *Design of an Application-Cooperative Management System for Wireless Sensor Networks.* In Proceeedings of the Second European Workshop on Wireless Sensor Networks, pages 121 – 132, Istanbul, Turkey, 2005.

[Tourancheau 09] B. Tourancheau, Y. Mazzer, V. Gavan, F. Kuznik & G. Krauss. *Software calibration of wirelessly networked sensors.* In Proceedings - 2009 3rd International Conference on Sensor Technologies and Applications, SENSORCOMM 2009, 2009 3rd International Conference on Sensor Technologies and Applications, SENSORCOMM 2009, pages 314–319, Athens, Glyfada, 2009.

[Wang 06] N. Wang, N. Zhang & M. Wang. *Wireless sensors in agriculture and food industry–Recent development and future perspective.* In Computers and Electronics in Agriculture, volume 50, pages 1–14, 2006.

[Welch 84] T. A. Welch. *A Technique for High-Performance Data Compression.* In Computer, volume 17, pages 8–19, 1984.

[Werner-Allen 05] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees & M. Welsh. *Monitoring volcanic eruptions with a wireless sensor network.* In Proceeedings of the Second European Workshop on Wireless Sensor Networks, pages 108–120, Boston, MA, USA, 2005.

[Winter 09] T. Winter, P. Thubert, ROLL Design Team & IETF ROLL WG. *RPL : IPv6 Routing Protocol for Low power and Lossy Networks.* In draft-ietf-roll-rpl-05, pages 1–80, 2009.

[Yao 02] Y. Yao & J. Gehrke. *The cougar approach to in-network query processing in sensor networks.* In SIGMOD Rec., volume 31, pages 9–18, 2002.

[Yen Yang 08] Lim Yen Yang, M. Messina, F. Kargl, L. Ganguli, M. Fischer & T. Tsang. *SNMP Proxy for Wireless Sensor Network.* In Information

Technology : New Generations, 2008. ITNG 2008. Fifth International Conference on, pages 738–743, 2008.

[Zhou 04] H. Zhou, K. M. Hou, J. Ponsonnaille, L. Gineste, J. Coudon, G. De Sousa, C. De Vaulx, J. J. Li, P. Chanais, R. Aufrère, A. Amamra & J. P. Chanet. *Remote Continuous Cardiac Arrhythmias Detection and Monitoring.* In Krzysztof Zielinski et David Ingram Mariusz Duplaga, editeur, Transformation of Healthcare with Information Technologies, Studies in Health Technology and Informatics, volume 105, pages 112–120. IOS Press, 2004.

[Zhou 06] H. Zhou, G. De Sousa, J.P. Chanet, K.M. Hou, J.J. Li, C. De Vaulx & M. Kara. *An Intelligent Wireless Bus-Station System Dedicated to Disabled, Wheelchair and Blind Passengers.* In IET International Conference on Wireless, Mobile and Multimedia Networks ICWMMN 2006, 4 pages, Hangzhou, China, 2006.

$\sim \star \sim \star \sim \star \sim$

Résumé

Les évolutions technologiques des dernières années ont permis de mettre en évidence un nouveau domaine de recherche : les Réseaux de Capteurs Sans Fil (RCSFs). Les RCSFs sont basés sur des systèmes embarqués à fortes contraintes de ressources telles que l'énergie, la puissance de calcul et la mémoire. Leurs domaines d'application sont vastes allant notamment de la collecte de données environnementales à la surveillance d'infrastructures en passant par l'aide aux personnes.

Une application type contient de quelques dizaines à plusieurs milliers de capteurs sans fil (nœuds). Le challenge que cette thèse se propose de relever est de fournir des méthodes simples et peu intrusives pour administrer ces réseaux. L'objectif est de répondre à ce besoin en minimisant l'impact sur le fonctionnement des nœuds et le trafic du réseau ainsi que le coût énergétique. La solution proposée - *LiveNode Non invasive, Context aware, and Modular management (LiveNCM)* - fournit des méthodes permettant de connaître l'état des nœuds et d'interagir avec eux en prenant en compte le contexte applicatif et en utilisant des estimateurs pour minimiser les échanges.

Déployé sur une application de collecte de données environnementales, LiveNCM propose un outil de supervision basé sur une extension du protocole SNMP. Les résultats obtenus notamment sur les gains énergétiques sont importants avec une autonomie du réseau augmentée de plus de 50% pour certaines grandeurs physiques observées dans un contexte spécifique. De plus, en minimisant les échanges dans le réseau, LiveNCM limite les collisions et les goulots d'étranglement qui peuvent apparaître sur le serveur de collecte des données et entre les nœuds.

Mots-clés : *Réseaux de capteurs sans fil, protocole SNMP, diagnostic indirect, non invasif, LiveNCM*

Abstract

Thanks to the advanced in wireless access medium, MEMS and VLSI technologies in the last decade, a new research area, which is considered as one of the key technology of the 21st century is emerged: Wireless Sensors Networks (WSNs). WSN is a high resource constraints embedded systems such as energy and processing power. Their application fields are huge, which may be applied from environmental data collection to disable people assistance.

A typical application can contain about ten or thousand of wireless sensors nodes. The challenge of this thesis is to provide a simple and intrusive-less technique to manage efficiently WSNs. To meet these requirements, it is important to minimize the impacts of WSN administration tasks on the operation of every sensor node and network traffic. Thus, we proposed a new technique named *LiveNode Non invasive, Context-aware, and Modular management (LiveNCM)*. LiveNCM is a SNMP like WSN management tool. The main features of LiveNCM are to provide methods, which enable to know every node status by taking into account the application context and by implementing sample data estimators to minimise message exchanges. Note that, the wireless message exchanges are energy consuming, so by minimizing message exchanges, we increase significantly the WSN lifetime comparing with the traditional techniques ones.

The results obtained from a real application: environmental data collection, shown that LiveNCM is energy efficient because it enables to double the WSN lifetime. Moreover, by minimizing the message exchanges in the network, LiveNCM limits message sending collisions and network traffics. Thus indirectly, LiveNCM improves significantly the QoS of WSN.

Keywords: *Wireless sensor networks, SNMP protocol, indirect diagnosis, non-invasive, LiveNCM*