Partitionnement et Geocasting dans les Réseaux Mobiles Ad hoc et Collecte des Données dans les Réseaux de Capteurs

Idrissa Sow

Université de Picardie Jules Verne

N° attribué par la bibliothèque

THÈSE

pour obtenir le grade de

Docteur de l'Université de Picardie Jules Verne

Spécialité : **Informatique**

préparée au laboratoire : **MIS**

dans le cadre de l'École Doctorale **Sciences et Santé**

présentée par

M. Idrissa SOW

Partitionnement et Geocasting dans les Réseaux Mobiles Ad Hoc et Collecte des Données dans les Réseaux de Capteurs Sans fil

Soutenance prévue le 4 juin 2009 devant le jury composé de:

M. Khaldoun AL-AGHA	Prof. Université Paris-Sud 11	Président du jury
M. Mohamed NAIMI	Prof. Université de Cergy Pontoise	Rapporteur
Mme. Véronique VÈQUE	Prof. Université Paris-Sud 11	Rapporteur
Mme. Houda LABIOD	Maître de Conf. Télécom ParisTech	Examinatrice
M. Jean Frédéric MYOUPO	Prof. UPJV, Amiens	Directeur de thèse

Remerciements

Je tiens à remercier Vénonique VÈQUE et Mohamed NAIMI pour avoir consacré un peu de leur précieux temps pour rapporter ce travail. Je tiens à exprimer ma gratitude à Khaldoun AL-AGHA et Houda LABIOD pour leur amabilité à participer à ce jury.

Je tiens à remercier particulièrement Jean Frédéric MYOUPO qui a dirigé mes travaux de recherche et qui m'a soutenu pendant ces trois années. Je tiens aussi à remercier Jean François LALANDE, Jeremy BRIFFAUT, Christian TOINARD, Martial SZPIEG et Patrice CLEMENTE de l'École Nationale Supérieure d'ingénieur de Bourges (ENSIB) pour m'avoir permis de mener à bien ma thèse durant mes années ATER dans les meilleures conditions possibles.

Je tiens à remercier en particulier mes collègues de bureau Yoan DIEUDONÉ et Gary HARDY avec qui j'ai partagé des bons moments.

Je remercie également tous les membres du laboratoire MIS anciennement LaRIA, avec qui j'ai eu le plaisir de travailler pendant ces trois années. Je ne peux que garder un très bon souvenir de vous tous.

Sans oublier le support inconditionnel de ma femme et de mes proches.

Résumé

L'essor des technologies sans fil, offre aujourd'hui de nouvelles perspectives dans le domaine des télécommunications. L'évolution récente de moyens de communications sans fil permet la manipulation de l'information à travers des unités de calculs portables qui ont des caractéristiques particulières (une faible capacité de stockage, une source d'énergie autonome...) et accèdent au réseau à travers une interface de communication sans fil. Un réseau mobile ad hoc consiste en une grande population relativement dense d'unités mobiles qui se déplacent dans un environnement quelconque et dont le seul moyen de communication est l'utilisation des interfaces sans fil. Du coté des réseaux ad hoc on trouve également les réseaux des capteurs avec des propriétés particulières. Les capteurs sont des dispositifs ayant les particularités suivantes : (1) capacité de capturer des données relatives à l'environnement où ils sont physiquement placés et les convertir en signaux électriques. Les données récupérées peuvent être de nature différente et la manière d'obtenir ces données est susceptible de varier. (2) capacité d'effectuer un traitement sur ces données récupérées .(3) capacité d'échanger ces données avec d'autres dispositifs ou unités mobiles.

Contrairement aux réseaux basés sur la communication cellulaire : aucune administration centralisée n'est disponible ; ce sont les hôtes mobiles elles-mêmes qui, forment d'une manière ad hoc une interface du réseau. Aucune supposition ou limitation n'est faite sur la taille du réseau ; le réseau peut contenir des centaines ou des milliers d'unités mobiles. Étant donné les propriétés de ces réseaux le routage devient alors un défi et nécessite une restructuration de ses différentes composantes qui sont le routage (ou adressage), le positionnement, l'acheminement des messages entre nœuds communicants.

Durant cette thèse nous nous sommes intéressés implicitement au routage en proposant une structure hiérarchique ou structure en clusters de façon à simuler une sorte de dorsale constituée de nœuds ou terminaux plus *adaptés* que d'autres. La mise en place de cette structure se base sur les paramètres du réseau en question. La dorsale ainsi constituée permet un routage hiérarchique qui allège considérablement les tables de routage des nœuds. Nous avons proposé une approche de localisation sans GPS, *L-Libre* qui vise à procurer une information de position à l'ensemble des nœuds du réseau. Cette information de position est vitale pour les protocoles de routage géographiques mais aussi pour les réseaux de capteurs où l'on a souvent bien besoin de connaître la source (les capteurs origines) des informations reçues, ce qui est d'ailleurs le cas de notre algorithme de poursuite de cible (tracking) proposé dans cette thèse. Nous nous sommes également intéressé à un mode de transmission d'information appelé *geocasting* (ou diffusion géographique) qui consiste à trans-

mettre des informations avec garantie de livraison non pas à l'ensemble des unités du réseau mais à un groupe particulier de nœuds situés dans une région d'intérêt communément appelé *région multicast*.

Un autre point important est la connectivité des nœuds. La mobilité des nœuds est souvent source de déconnexion du réseau en des ensembles disjoints connexes. Notre algorithme de maintenance de connectivité vise à déterminer les nœuds qui peuvent être à l'origine de ce problème afin d'adapter leurs comportements.

Et finalement, nous avons considéré une architecture virtuelle de réseau de capteurs anonymes dans laquelle certains clusters peuvent être vides. Les capteurs étant des dispositifs de capacités très limitées il devient alors indispensable de trouver une approche permettant d'organiser ou de faire collaborer à moindre coût ces capteurs dans un but bien précis. Le nœud sink[1] (ou nœud puits) est le seul équipement qui dispose d'assez d'énergie et de puissance de transmission pour atteindre l'ensemble des capteurs répartis dans la région d'intérêt. Une stratégie de gestion de ou des antennes du nœud sink nous a permis de proposer une méthode de localisation sans inonder le réseau par échange de messages d'informations de position. Il nous a également permis d'esquisser une structure en grappes offrant un support de routage simple et efficace. Les données ainsi recueillies par les capteurs peuvent être acheminées selon un modèle de communication centralisé ou distribué défini à l'avance par le sink.

Mots Clés

Réseaux sans fil multi-sauts, réseaux ad hoc, réseaux de capteurs, partitionnement, positionnement, geocasting, poursuite de cible, connectivité

[1] nœud servant d'interface entre le réseau et son utilisateur final

Abstract

The rise of wireless technologies, today offers new telecommunications opportunities. The recent technology wireless communication capabilities and allows the manipulation of information across small devices that have special features (a low storage capacity, an autonomous source of energy ...) and access to the network interface through a wireless communication liks. A mobile ad hoc network is a large number of devices capable to move in any environment and whose only use means of communication by using their wireless links. On the side of ad hoc networks are also networks of sensors with specific properties. The sensors are devices with some particularity : (1) ability to collect data on the environment where they are physically located and convert them into electrical signals. The data collected can be different in nature and the way to collect such data is likely to vary. (2) ability to make treatment on these data collected. (3) ability to exchange data with other devices or mobile units.

Unlike networks based on cellular communication : no centralized administration is available the mobile hosts form themselves an ad hoc network interface. No assumption is made or limitation on the size of the network, the network may contain hundreds or thousands of mobile devices. Since the properties of these networks routing becomes a challenge and requires a restructuring of its various components that are routing (or address), positioning, routing messages between communicating nodes.

During this thesis we are interested in routing implicitly suggesting a hierarchical structure or clustered structure so as to simulate a kind of backbone consists of nodes or more adapted than others. The establishment of this structure is discussed on the parameters of the network. Constituted the backbone allows routing hiérachique to ease the routing tables of nodes. We have proposed a location method without GPS, *L-Libre*, which aims to provide position information to all nodes in the network. This position information is vital for routing geographical protocols but also for sensor networks where it was often necessary to know the source (sensors origins) information received. We are also interested in a mode of transmission called *geoscasting* which is to transmit information with guarantee delivery not to all nodes of the network but to a particular group of nodes located in a region of interest commonly called *multicast group*.

Another important point of this thesis is the node's mobility. Due to the mobility patterns, node can be deparated into patitions and reconnected a number of times. Our maintenance connectivity algorithm aims to determine the nodes that can cause

this problem in order to adapt theirs behaviors.

The last point of this thesis we have considered a virtual sensors anonymous network architecture in which some clusters may be empty. The sensors are small devices with limited capacities, it becomes necessary to find an approach to organize at lower cost sensors in a specific purpose. The sink node is the only equipment that has enough energy and transmission power that can reach all the sensors distributed throughout the region of interest. A management strategy or branches of the sink node has enabled us to propose a method of locating without flooding the network with information positions messages exchange. It also helped us to tring a clustered structure offering an efficient routing. The data collected by the sensors can deliver as a centralized or distributed communication model defined in advance by the sink node.

Table des matières

Chapitre 1

Introduction

1 Introduction

Avec l'émergence des dispositifs sans fil, de multiples recherches ont été faites dans les communications mobiles et sans fil. Ces nouvelles recherches se concentrent en particulier sur les façons d'employer et de traiter simultanément tous ces dispositifs hétérogènes afin de les organiser en réseau n'exigeant aucune configuration ni d'infrastructure préétablie. De tels réseaux sont appelés *réseaux ad hoc* ou *réseaux spontanés*. Pour rendre ces réseaux autonomes, chaque nœud ou composant doit collaborer avec ses voisins pour échanger des informations. Tous les terminaux ou nœuds de ce type de réseau peuvent communiquer soit par un lien direct (s'ils sont suffisamment proches les uns des autres) soit grâce à la coopération d'un ou plusieurs autres membres du réseau. La difficulté dans ce type de réseau est que tout terminal agira à la fois en tant que source et en tant que relais selon la situation. En conséquence, la plupart des tâches *supposées faciles* sur des réseaux classiques deviennent nettement plus problématiques (diffusion, routage, etc.).

Compte tenu de la mobilité imprévisible des nœuds, la topologie d'un réseau ad hoc est instable et soumise à de fréquents changements. Router dans de telles conditions devient une tâche complexe. De plus le passage à l'échelle dans les réseaux ad hoc semble être une question préoccupante quand le nombre de nœuds mobiles augmente. Le délai pour la mise-au-point de tels réseaux peut sembler quelque peu pharaonique mais leurs applications pratiques seraient telles qu'il ne faut pas mesurer ses efforts. Cela inclut des applications militaires (opérations spéciales, scénarios de champs de bataille, etc.), catastrophes naturelles (incendies, tremblement de terre, inondation, etc), applications évenementielles (installation d'un réseau provisoire dans le cadre par exemple d'exposition ou d'un grand rassemblement,etc).

Une façon pratique de contourner les difficultés de routage liées aux changementx de topologie dûs à la mobilité, à la taille du réseau et aux contraintes d'énergie serait de pouvoir construire une structure hiérarchique au-dessus du réseau de façon à simuler une sorte de dorsale constituée de nœuds ou terminaux plus *adaptés* que d'autres. Cette démarche a déjà prouvé son efficacité dans le passé et constitue un problème bien connu et largement étudié dans les systèmes distribués classiques. Elle a notamment aidé à la résolution de plusieurs problèmes comme la minimisation de

l'espace de stockage pour la communication d'informations (par exemple des tables de routages, etc.). Ainsi l'amélioration de la bande passante est effective ainsi que la distribution de ressources. La notion d'organisation en clusters a été utilisée pour les réseaux ad hoc depuis leur apparition. Une architecture complètement distribuée en *clusters* est proposée principalement dans le but d'établir un routage hiérarchique et de démontrer la capacité d'adaptation de ces réseaux aux changements de connectivité.

La mobilité des nœuds dans ce type de réseau est certes pratique mais il n'est pas sans conséquence. Le problème classique causé par les nœuds en mouvement dans un réseau ad hoc est la décomposition de la structure en des ensembles disjoints. Prévoir ces partitions ou sous-ensembles qui peuvent scinder le réseau est d'une grande utilité dans un environnement ad hoc mobile. En effet, être conscient d'un avenir de déconnexion du réseau peut aider à assurer une meilleure qualité de service par l'adaptation du comportemet des applications.

Une application très répandue des réseaux ad hoc est les réseaux de capteurs Les capteurs sont des petits dispositifs dont le but est de relever des informations concernant l'environnement dans lequel ils sont déployés et de les communiquer entre eux ou vers un nœud puits (appelé sink en anglais). Les réseaux de capteurs sont en général constitués de centaines, voire de milliers de nœuds et sont parfois conçus pour servir une fois qu'ils sont déployés dans des milieux hostiles (volcans, océans, champs de bataille). Que les capteurs soient mobiles ou non, l'information de position devient nécessaire dans plusieurs situations. La position des capteurs est devenue une information à grande valeur ajoutée. De nombreuses applications ou services dépendent et se servent de la position des capteurs.

Les réseaux de capteurs peuvent être destinés à plusieurs applications [15], parmi lesquelles :

⇒ Applications militaires : dans ce domaine, les capteurs peuvent être utilisés pour la surveillance, la détection d'intrusion, la détection de substances dangereuses, la communication, la reconnaissance et le ciblage.

⇒ Applications commerciales : partant du principe que toute application peut avoir un intérêt commercial, il apparaît clairement que les réseaux de capteurs ont leur place dans ce domaine. Parmi les applications, nous pouvons citer le contrôle environnemental des bâtiments (temperature, taux d'humidité, etc.) qui permet une meilleure gestion des ressources et participe ainsi à la réduction des frais. Nous pouvons également citer les musées scientifiques où les capteurs permettent une interactivité entre les visiteurs et le musée leur permettant un apprentissage plus rapide.

⇒ Applications environnementales : les capteurs peuvent être utilisés pour traquer les mouvements d'animaux ainsi que pour surveiller les conditions d'environnement qui affectent les récoltes, stocks et autres systèmes agricoles. Ils sont également d'une forte utilité dans la détection de catastrophes naturelles telles que les inondations, feux de forêts et tremblements de terres.

⇒ Applications médicales : parmi les applications médicales, nous pouvons citer celles qui permettent de surveiller les données physiologiques des patients et les taux de médicaments qui leur sont administrés. Les capteurs peuvent également aider à localiser les patients et médecins au sein d'un hôpital par exemple.

⇒ Applications architecturales : dans ce contexte, les capteurs transforment les bâtiments en environnements intelligents. Un tel environnement est capable, par exemple, de reconnaître des personnes, interpréter leurs actions et y réagir.

1.1 Problèmes abordés et contributions

Durant cette thèse nous nous sommes consacrés :

- au partitionnement ou clusterisation dans les réseaux mobiles ad hoc. Il s'agit d'un algorithme distribué de partitionnement qui utilise une heuristique de poids à la différence des heuristiques de *plus petit identifiant* [36] et du *plus grand degré* [37]. Le poids est ici un *n-uplet* de certains paramètres du réseau.

- à la maintenance de la connectivité dans les réseaux ad hoc mobiles. Le réseau ad hoc par définition engendre des propriétés de connectivité qui demeurent le plus souvent critiques. Une décomposition du réseau ad hoc en plusieurs composantes connexes est liée à la mobilité non contrôlée des nœuds. La maintenance de la connectivité que nous proposons consiste pour chaque nœud à détecter à l'avance les liens de communications qui peuvent diviser son voisinage en un ou plusieurs ensembles connexes mais aussi dans la mesure du possible de les éviter ou de les retarder.

- à la collecte et à la fusion des données dans une architecture virtuelle de réseau de capteurs qui se compose d'un système de coordonnées dynamiques, d'une structure en grappes de capteurs et d'un modèle de communications pour l'acheminement des données collectées par les capteurs. Cette architecture sert aussi de support pour une répartition en parallèle de plusieurs applications sur un même réseau de capteurs :

- à une géo-localisation centralisée au nœud sink basée sur une stratégie d'utilisation de l'antenne omnidirectionnelle de ce dernier.

- à une diffusion géographique (ou geocasting) avec garantie de livraison. Le geocasting est un procédé qui consiste à envoyer des données à une poignée de capteurs situés dans une région d'intérêt (région géographique) appelée région multicast. Le terme multicast est utilisé pour désigner une méthode de diffusion de l'information d'un émetteur vers un groupe.

- à un travail collaboratif pour la détection et la poursuite de cible dans une architecture clusterisée de capteurs.

1.2 Plan du manuscrit

La suite de cette thèse contient cinq chapitres. Le deuxième chapitre explore les typographies des réseaux sans fil. Dans chaque chapitre nous présenterons une étude succinte des solutions existantes avant d'apporter notre contribution. Au chapitre 3 nous présentons les détails de notre algorithme de partitionnement. Nous exposons au chapitre 4 notre stratégie de localisation sans GPS centré au nœud sink (ou source) et le geocasting avec garantie de livraison. Le chapitre 5 traite de la maintenance de la connectivité dans les environnements mobiles ad hoc de manière générale et de la poursuite de cible (tracking) dans les réseaux des capteurs en particulier. Nous présentons au chapitre 6 la collecte et la fusion des données dans une architecture virtuelle de réseau de capteurs. Et enfin le chapitre 7 clôt cette thèse et décrit les travaux futurs.

Chapitre 2

Les types de réseaux sans fil

Les réseaux mobiles sont en plein développement du fait de la flexibilité de leur interface, qui permet à un utilisateur de changer de place tout en restant connecté. Les communications entre équipements terminaux peuvent s'effectuer directement ou par le biais de stations de base, appelées points d'accès, ou AP (Access Point). Les communications entre points d'accès peuvent être hertziennes ou par câble. Les débits de ces réseaux se comptent en dizaines de mégabits par seconde. Plusieurs gammes de produits sont actuellement commercialisées, mais la normalisation pourrait encore modifier les choses. Les groupes de travail qui se chargent de cette normalisation proviennent de l'IEEE aux États-Unis et de L'ETSI en Europe. La figure 2.1 décrit les différentes catégories de réseaux suivant leur étendue et la figure 2.2 montre les normes existantes.

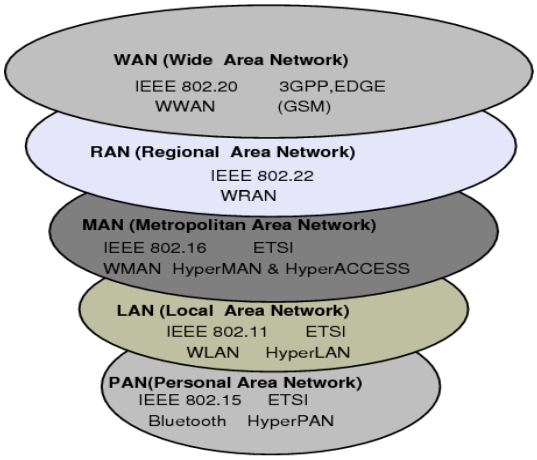


Fig. 2.1 – Catégories de réseaux sans fil.

Les principales normes sont IEEE 802.15, pour les petits réseaux personnels d'une dizaine de mètres de portée, IEEE 802.11, ou Wi-Fi, pour les réseaux locaux sans fil (WLAN) IEEE 802.16, pour les réseaux métropolitains sans fil (WMAN) atteignant plus de dix kilomètres, IEEE 802.22, pour les réseaux régionaux sans fil (WRAN), et IEEE 802.20, pour les réseaux étendus sans fil (WWAN), qui correspondent aux solutions cellulaires permettant de couvrir un pays. Pour cette catégorie de réseaux, nous avons retenu la proposition, IEEE 802.20, qui peut être considérée comme une des solutions multimédias à très haut débit concurrentes de la future quatrième génération de réseaux de mobiles.

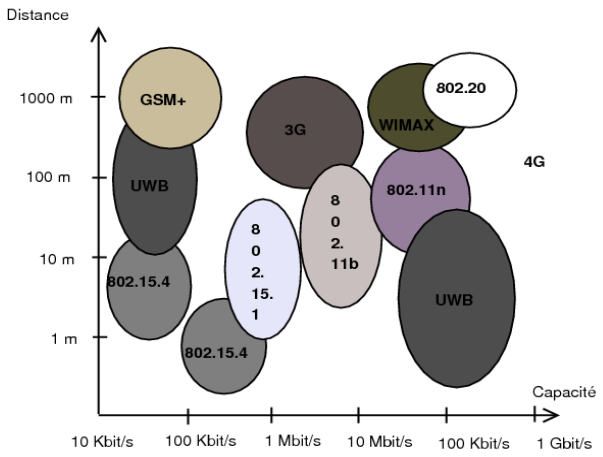


FIG. 2.2 – Principales normes des réseaux sans fil

Dans le groupe IEEE 802.15, trois sous-groupes normalisent des gammes de produits en parallèle :

◇ IEEE 802.15.1, le plus connu, prend en charge la norme Bluetooth, aujourd'hui largement commercialisée. La version 3.0 utilise l'interface radio décrite dans IEEE 802.15.3, ce qui procure à Bluetooth une nouvelle jeunesse, avec un débit de 480 Mbit/s.

◇ IEEE.15.3 définit la norme UWB (Ultra-Wide Band), qui met en œuvre une technologie très spéciale, caractérisée par l'émission à une puissance extrêmement faible, sous le bruit ambiant, mais sur pratiquement l'ensemble du spectre radio (entre 3,1 et 10,6 GHz). Le débit est de 480Mbit/s sur une portée de 3m et décroît à environ 120 Mbit/s à une dizaine de mètres.

◇ IEEE 802.15.4 s'occupe de la norme ZigBee, qui a pour objectif de promouvoir une puce offrant un débit relativement faible mais à un coût très faible. ZigBee est avant tout normalisé pour le passage des commandes plutôt que des données. Cependant, une version sortie en 2007 propose d'utiliser l'UWB et offre donc malgré tout un débit important.

Du côté de la norme IEEE 802.11, dont les produits sont nommés Wi-Fi (Wireless-Fidelity), il existe aujourd'hui trois propositions, dont les débits sont de 11 Mbit/s (IEEE 802.11b) et 54 Mbit/s (IEEE 802.11 a et g). Une quatrième proposition, provenant des travaux du groupe IEEE 802.11n, permet d'augmenter fortement le débit, avec une centaine de mégabits par seconde au mieux en débit réel. Les fréquences utilisées se placent dans la bande 2,4-2,4835 MHz pour les extensions b et g dans la bande 5,15-5,35 MHz pour 802.11a.

Les réseaux WiMAX IEEE 802.16 visent à remplacer les modems ADSL, que l'on trouve sur les réseaux téléphoniques fixes, pour donner à l'utilisateur final des débits du même ordre de grandeur que l'ADSL, jusqu'à plusieurs Mbits/s. Ces réseaux forment ce que l'on appelle la boucle locale radio. Plusieurs normes sont proposées

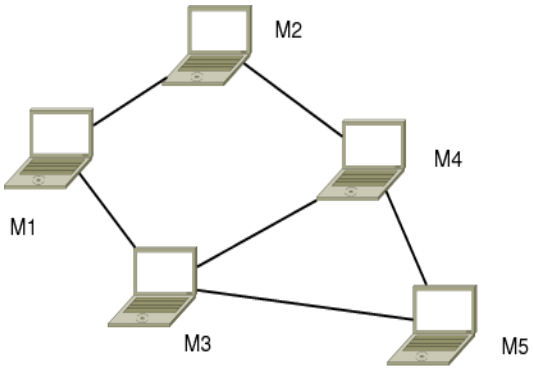


FIG. 2.3 – réseau ad-hoc.

suivant la fréquence utilisée. Un consortium s'est mis en place pour développer les applications de cette norme sous le nom de WiMAX. Deux versions sont commercialisées, l'une fixe, dont l'objectif est clairement de remplacer l'ADSL dans les zones rurales, l'autre mobile, permettant d'avoir un modem ADSL dans sa poche toujours connecté.

Les réseaux régionaux sont étudiés par l'IEEE 802.22. Le rayon de la cellule peut atteindre 50 kilomètres pour les gammes de fréquences en dessous de 1 GHz. La distance potentielle du terminal étant importante, le débit montant est assez limité. En revanche sur la bande descendante, 4 Mbit/s sont disponibles. L'application de base est la télévision interactive ou les jeux vidéo interactifs.

En ce qui concerne les WAN (Wide Area Network), c'est plutôt l'interconnexion des réseaux précédents qui les supporte. Pour cela, il fallait définir une norme d'interconnexion, qui a été apportée par les spécifications du groupe IEEE 802.21. On peut aussi classer dans cette catégorie la norme IEEE 802.20, qui correspond à des cellules cohérentes et permet les accès large bande.

1 Les réseaux ad hoc

Une autre grande catégorie de réseaux sans fil est constituée par les réseaux ad-hoc [47], réseaux capables de s'organiser sans infrastructure définie préalablement. Ces dernières acceptent de jouer le rôle de routeur pour permettre le passage de l'information d'un terminal vers un autre, sans que ces terminaux ne soient reliés directement. Un réseau ad-hoc est illustré à la figure 2.3. Contrairement aux apparences, les réseaux ad-hoc datent de la dernière décennie. Ils visent à réaliser un environnement de communication qui se déploie sans autre infrastructure que les mobiles eux-mêmes. En d'autres termes, les mobiles peuvent jouer le rôle de passerelle pour permettre une communication d'un mobile à un autre. Deux mobiles trop éloignés l'un de l'autre pour communiquer directement doivent trouver un mobile intermédiaire capable de jouer de rôle de relais.

La difficulté majeure engendrée par ce type de réseau provient de la définition même de la topologie du réseau : comment déterminer quels sont les nœuds voisins

et comment aller d'un nœud vers un autre nœud ? Deux solutions extrêmes peuvent être comparées. La première est celle d'un réseau ad hoc dans lequel tous les nœuds peuvent communiquer avec tous les autres, impliquant une longue portée des émetteurs. Dans la seconde solution, au contraire, la portée hertzienne est la plus courte possible : pour effectuer une communication entre les deux nœuds, il faut généralement passer par plusieurs nœuds intermédiaires. L'avantage de la première solution est la sécurité de la transmission puisqu'on peut aller directement de l'émetteur au récepteur, sans dépendre d'un équipement intermédiaire. Le débit du réseau est minimal, les fréquences ne pouvant être réutilisées. Dans le second cas, si un terminal tombe en panne ou est éteint, le réseau peut se couper en deux sous-réseaux distincts, sans communication de l'un à l'autre. Bien évidemment, dans ce cas, le débit global est optimisé, puisqu'il peut y avoir une forte réutilisation des fréquences.

Les techniques d'accès sont du même type que dans les réseaux de mobiles. Cependant, du fait que tous les portables jouent un rôle de BSS et qu'ils sont eux-mêmes mobiles, de nouvelles propriétés doivent être apportées à la gestion des adresses des utilisateurs et au contrôle du routage. La solution développée pour les réseaux ad-hoc prend pour fondement l'environnement IP. Les mobiles qui jouent le rôle de passerelle implémentent un routeur dans leurs circuits, de telle sorte que les problèmes posés reviennent essentiellement à des problèmes de routage dans Internet, la mobilité étant gérée par le protocole IP Mobile.

Les avantages des réseaux ad-hoc sont leurs extensions très simples, leur couverture physique et leur coût. Toutefois, pour en bénéficier pleinement, un certain nombre d'écueils sont à surmonter, telles la qualité de service et la sécurité, du fait de la mobilité des nœuds. La topologie variable et l'absence d'infrastructure des réseaux ad hoc posent des problèmes pour mettre en place une politique de gestion de qualité de service comme dans les réseaux classiques. Dans les réseaux ad hoc, la qualité de service [27] n'est pas du tout appropriée car celle-ci dépend de la qualité du réseau. Par conséquent, se pose la question de savoir comment appliquer une politique de QoS dans un réseau où l'on note l'absence d'infrastructure, une grande mobilité des nœuds et des ressources limitées ? A présent, même si des solutions comme l'application des modèles en couches existent, beaucoup reste à faire.

Les réseaux sans fil en particulier les réseaux ad hoc sont généralement plus sensibles aux menaces physiques et aux attaques que les réseaux classiques. Les possibilités accrues d'attaque par écoute, intrusion, usurpation d'identité par déni de service, embrouillement des bandes passantes, doivent être étudiées avec la plus grande attention. Actuellement, l'un des problèmes majeurs lié à la sécurité dans les réseaux ad hoc est la confidentialité des données puisque celles-ci transitent par des nœuds intermédiaires avant d'atteindre leur destination. D'où une utilité accrue pour assurer la confidentialité. Actuellement, la sécurisation des MANET s'appuie sur des mécanismes de sécurité du lien radio (WEP) et des échanges IP.

MANET est le groupe de travail de l'IETF qui se préoccupe de la normalisation des protocoles ad-hoc fonctionnant sous IP. Ce groupe s'est appuyé sur les protocoles classiques d'Internet et les a perfectionnés pour qu'ils puissent fonctionner avec des

routeurs mobiles. Deux grandes familles de protocoles ont été définies : les protocoles *réactifs* et les protocoles *proactifs* :

- ◇ **Protocoles réactifs**. Les terminaux ne maintiennent pas de table de routage mais s'en préoccupent lorsqu'une émission est à effectuer. Dans ce cas, on se sert essentiellement de techniques d'inondation pour répertorier les mobiles pouvant participer à la transmission.

- ◇ **Protocoles proactifs**. Les mobiles cherchent à maintenir une table de routage cohérente, même en l'absence de communication.

Les réseaux ad-hoc sont utiles dans de nombreux cas de figure. Ils permettent de mettre en place des réseaux dans un laps de temps restreint, en cas, par exemple, de tremblement de terre ou pour un meeting avec un très grand nombre de participants. Une autre possibilité est d'étendre l'accès à une cellule d'un réseau sans fil comme le Wi-Fi. Comme illustré à la figure 2.4, un terminal situé hors d'une cellule peut se connecter à une machine d'un autre utilisateur se trouvant dans la zone de couverture de la cellule. Ce dernier sert de routeur intermédiaire pour accéder à l'antenne de la cellule. Les réseaux ad-hoc posent de nombreux problèmes du fait de la mobilité

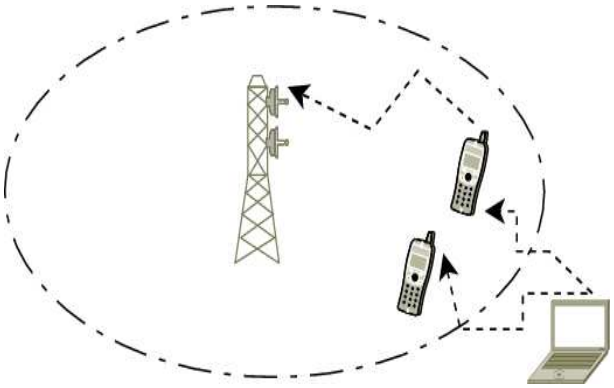


FIG. 2.4 – Extension de couverture par un réseau ad-hoc.

de tous les équipements. Le principal d'entre eux est le routage nécessaire pour transférer les paquets d'un point à un autre point du réseau. L'un des objectifs du groupe MANET est de proposer une solution à ce problème. Pour le moment, quatre grandes propositions ont vu le jour, deux de type réactif et deux de type proactif. Parmi les autres problèmes, nous trouvons la sécurité, la qualité de service et la gestion de la mobilité en cours de communication.

1.1 Le routage

Le routage est l'élément primordial d'un réseau ad-hoc. Il faut un logiciel de routage dans chaque nœud du réseau pour gérer le transfert des paquets IP. La solution la plus simple est évidemment d'avoir un routage direct, comme celui illustré à la figure 2.5 dans lequel chaque station du réseau peut atteindre directement une autre station, sans passer par un intermédiaire (on parle de réseau à un saut). Ce cas

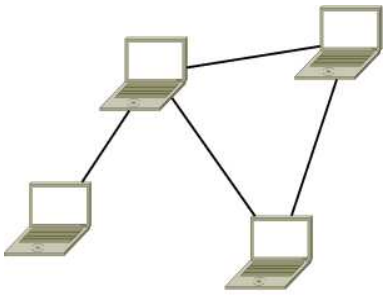


FIG. 2.6 – Communication indirecte entre station.

le plus simple correspondant à une petite cellule, d'un diamètre inférieure à 100m, comme dans un réseau 802.11 en mode ad-hoc.

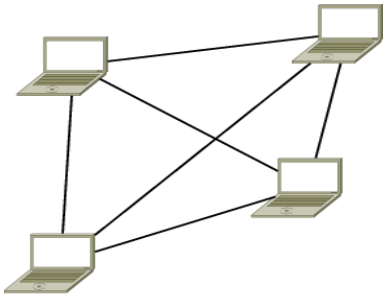


FIG. 2.5 – Communication directe entre station.

Le cas classique du routage dans un réseau ad-hoc consiste à transiter par des nœuds intermédiaires. Ces derniers doivent posséder une table de routage apte à diriger le paquet vers le destinataire. Toute la stratégie d'un réseau ad-hoc consiste à optimiser les tables de routage par des mises à jour plus ou moins régulières. Si les mises à jour sont très régulières, cela risque de surcharger le réseau. Cette solution présente toutefois l'avantage de maintenir des tables à jour et donc de permettre un routage rapide des paquets. Une mise à jour uniquement lors de l'arrivée d'un nouveau flot restreint la charge circulant dans le réseau mais décharge le réseau de nombreux flots de supervision. Il faut dans ce cas arriver à mettre en place des tables de routage susceptibles d'effectuer l'acheminement dans des temps acceptables. La figure 2.6 illustre la cas d'un réseau ad-hoc dans lequel, pour aller d'un nœud à un autre, il peut être nécessaire de traverser des nœuds intermédiaires.

Pour toutes ces raisons, les routes du réseau doivent être sans cesse modifiées, d'où l'éternelle question débattue à l'IETF : faut-il maintenir les tables de routage dans les nœuds mobiles d'un réseau ad-hoc ? En d'autres termes, vaut-il la peine de maintenir à jour des tables de routage qui changent sans arrêt ou n'est-il pas plus judicieux de déterminer la table de routage au dernier moment ?

Comme expliqué précédemment, les protocoles réactifs [9, 10, 25, 53] travaillent par inondation pour déterminer la meilleure route lorsqu'un flot de paquets est prêt à être émis. Il n'y a donc pas d'échange de paquets de contrôle en dehors de la supervision pour déterminer le chemin du flot. Le paquet de supervision qui est dif-

Métrique	Réactif	Proactif
Vecteur de distance	AODV	DSDV
Routage à la source	DSR	
État de lien		OLSR

TAB. 2.1 – Protocoles ad-hoc

fusé vers tous les nœuds voisins est de nouveau diffusé par les nœuds voisins jusqu'à atteindre le récepteur. Suivant la technique choisie, on peut se servir de la route déterminée par le premier paquet de supervision qui arrive au récepteur ou prévoir plusieurs routes en cas de problème sur la route principale.

Les protocoles proactifs [5,8,39] se comportent différemment. Les paquets de supervision sont émis sans arrêt dans le but de maintenir à jour la table de routage en ajoutant de nouvelles lignes et en supprimant certaines. Les tables de routage sont donc dynamiques et varient en fonction des paquets de supervison parvenant aux différents nœuds. Une difficulté consiste dans ce cas à calculer une table de routage qui soit compatible avec les tables de routage des différents nœuds de telle sorte qu'il n'y ait pas de boucle.

Une autre possibilité consiste à trouver un compromis entre les deux systèmes. Cela revient à calculer régulièrement des tables de routage tant que le réseau est peu chargé. De la sorte, les performances des utilisateurs en transit ne sont pas trop modifiées. Lorsque le trafic augmente, les mises à jour sont ralenties. Cette méthode simplifie la mise en place d'une table de routage réactive lorsqu'une demande parvient au réseau.

Les protocoles proposés à la normalisation du groupe MANET sont récapitulés au tableau 2.1. Différentes métriques peuvent être utilisées pour calculer la meilleure route :

- ◇ Les vecteurs de distances donnent un poids à chaque lien et additionnent les poids pour déterminer la meilleure route, qui correspond à celle du poids le plus faible.

- ◇ Le routage à la source permet de déterminer la meilleure route comme étant celle qui permet au paquet de supervision d'arriver le premier au destinataire.

- ◇ Les états de lien indiquent les liens qui sont intéressants à prendre et ceux qui le sont moins.

1.2 Routage AODV

AODV (Ad-hoc On Demand Distance Vector) [53] a été le premier protocole normalisé par le groupe MANET, juste avant OLSR. Il est du type réactif. Ce protocole peut gérer à la fois les routages unicast et multicast. Lorsqu'un flot de paquets est émis par un nœud, la première action est de déterminer la route par une technique

d'inondation. Pour cela, le paquet de requête de connexion mémorise les nœuds traversés lors de la diffusion. Lorsqu'un nœud intermédiaire reçoit une requête de connexion , il vérifie qu'il n'a pas déjà reçu une telle requête. Si la réponse est positive, un message est renvoyé vers l'émetteur pour indiquer l'abandon de cette route. Le premier message qui arrive au destinataire détermine la route à suivre. La complexité du processus de détermination de la route doit être simplifiée au maximum en évitant les diffusions inutiles. Pour cela, chaque requête de demande d'ouverture d'une route est numérotée, afin d'éviter les duplications, et possède un TTL qui limite le nombre de transmissions dans le réseau.

L'avantage d'AODV est de ne pas créer de trafic lorsqu'il n'y pas de message à transmettre. La détermination de la route est assez simple et n'implique que peu de calcul dans chaque nœud. Il est évident que les deux inconvénients majeurs résident dans le temps de mise en place de la route et de l'important trafic suscité pour mettre en place les routes.

1.3 Routage OLSR

Le protocole OLSR (Optimized Link State Routing) [8] est certainement le plus utilisé des protocoles de routage ad-hoc. Il est de type proactif. Pour éviter de transporter trop de paquets de supervision, OLSR s'appuie sur le concept de relais multipoints ou MPR (MultiPoint Relay). Les MPR sont des nœuds importants qui ont la particularité d'être les meilleurs points de passage pour atteindre l'ensemble des nœuds lors d'un processus d'inondation. L'état des liens n'étant envoyé que par les MPR, cela réduit d'autant les messages de supervision. La connaissance de ses voisins est obtenue par les paquets Hello qui sont émis en diffusion. Cela permet de déterminer quels sont les voisins et informations d'état de lien nécessaires pour l'algorithme de routage. Le message Hello permet également d'indiquer les MPR à ses voisins. Ces messages Hello ne sont destinés qu'aux nœuds voisins et ne peuvent être routés vers un destinataire à deux sauts.

2 Les réseaux maillés

Les réseaux maillés (meshed networks) [21] sont des réseaux ad-hoc dans lesquels les points de routage sont immobiles. Les clients sont rattachés par un réseau sans fil sur les points d'accès, et les points d'accès sont reliés entre eux par des liaisons sans fil.

L'avantage de ces réseaux est qu'ils peuvent couvrir une zone géographique importante, sans nécessiter de pose de câbles. Par exemple, sur un grand campus, les points d'accès peuvent se mettre sur les toits des différents bâtiments sans que l'architecte du réseau ait à se préoccuper de relier les points d'accès à un système câblé de type Ethernet. Plusieurs possibilités se font jour pour réaliser un réseau mesh :

- ◇ Utiliser la même fréquence que les terminaux, en considérant que les points d'accès sont traités comme des machines terminales. L'inconvénient est bien sûr d'utiliser la bande passante enlevée aux autres machines terminales. De plus, il faut faire attention que les deux points d'accès ne soient pas trop

éloignés et n'obligent l'émetteur et le récepteur à baisser leur vitesse. Cette solution est considérée comme la première génération de réseau mesh.

◇ Utiliser des fréquences différentes. Par exemple, un Wi-Fi 802.11b comportant trois fréquences disponibles, il est possible d'utiliser deux cartes de communication avec des fréquences différentes. L'inconvénient est bien sûr de perturber le plan de fréquences, surtout si le réseau est important et possède de nombreux points d'accès. Cette solution fait partie de la seconde génération de réseau mesh.

◇ Toujours dans la deuxième génération, le réseau mesh fait appel à une norme différente pour relier les points d'accès entre eux. Par exemple, un réseau mesh 802.11g peut utiliser la norme 802.11a pour interconnecter les points d'accès.

◇ On considère que la troisième génération utilise trois fréquences au total. Une pour connecter les clients,et deux pour interconnecter les points d'accès. Dans ce cas les connexions en amont et en aval d'un même nœud utilisent des fréquences différentes. Il faut généralement utiliser 802.11a qui possède jusqu'à huit fréquences différentes.

Les réseaux mesh posent des problèmes inédits aux réseaux sans fil, notamment les suivants : comment optimiser les batteries des points d'accès si ceux-ci ne sont pas reliés au courant électrique ? Comment optimiser le routage pour ne pas perturber le trafic utilisateur aux points d'accès, surtout s'ils sont déjà saturés ? Quelle densité de points d'accès faut-il utiliser, ce qui revient à se poser la question de la puissance des points d'accès ?

L'avantage de cette technologie est d'être capable de se reconfigurer facilement lorsqu'un point d'accès tombe en panne. Les clients peuvent se connecter à un autre point d'accès, quitte à augmenter légèrement la puissance des points d'accès voisins de celui en panne. Le problème principal est de gérer le routage. Ce dernier est traité dans les points d'accès qui ne sont pas des machines très puissantes, et il faut de ce fait éviter les points d'accès qui supportent beaucoup de trafic provenant des clients raccordés. De nombreuses propositions sont en discussion.

3 Les réseaux de capteurs

Un réseau de capteurs [22] se définit comme un ensemble de capteurs connectés entre eux, chaque capteur étant muni d'un émetteur-récepteur. Les réseaux de capteurs forment une nouvelle génération de réseaux aux propriétés spécifiques, qui n'entrent pas dans le cadre des architectures classiques.

La miniaturisation des capteurs pose des problèmes de communication et de ressources d'énergie. Il faut que le capteur soit suffisamment intelligent pour rassembler l'information requise et l'émettre à bon escient. De plus, le processeur du capteur ne doit pas être utilisé trop intensivement afin de consommer le moins d'énergie possible. Il doit donc incorporer des éléments réactifs plutôt que cognitifs. Enfin pour

assurer un bon débit, la portée des émetteurs-récepteurs doit être nécessairement faible, de l'ordre d'une dizaine de mètres. La mise en place d'un réseau de capteurs pose donc des problèmes de routage, de contrôle des erreurs et de gestion de l'alimentation.

Du point de vue de la communication, l'environnement des protocoles IP est trop lourd et engendre un débit trop important et une surconsommation. Les solutions qui ont été dérivées des réseaux de terrain, ou réseaux industriels à temps réel, présentent un meilleur compromis entre efficacité et énergie consommée. Comme les capteurs peuvent être diffusés par centaine au mètre carré, l'adressage IPv6 semble le plus probable. Pour le moment, les problèmes de sécurité et de qualité sont mis au second plan par rapport aux problèmes de consommation. Un champ de recherche important est en tout cas ouvert pour rendre les réseaux des capteurs efficaces et résistants. Les principaux standards radio concernent ZigBee. WiBee et 6lowPAN forment d'autre solutions. WiBee est une technologie très basse consommation d'une portée de 10m et d'un débit de 1Mbit/s. Les 6lowPAN (IPv6 over Low power Wireless Personal Area Networks) proviennent d'un groupe de travail de l'IETF. L'objectif est clairement de permettre la continuité du protocole IP vers des machines peu puissantes et avec une puissance électrique limitée.

L'utilisation de la norme IPv6 pour obtenir un très grand nombre d'adresses pour les immenses réseaux de capteurs pose problème. En effet les seize octets d'adresse de l'émetteur et les seize octets d'adresse du récepteur plus les champs obligatoires impliquent une mauvaise utilisation de la liaison radio pour transporter ces informations de supervision. Cela devient réellement problématique avec le peu d'énergie des capteurs. Ces réseaux de capteurs forment des réseaux mesh, et il leur faut un protocole de routage. Pour cette raison, les protocoles actuels sont beaucoup plus simple, avec des protocoles LOAD (6LowPAN AD-hoc Routing Protocol), une simplification d'AODV, DyMO-Low (Dynamic MANET On -demand for 6LowPAN), une simplification de DyMO du groupe de travail MANET, et Hi-Low (Hierarchical Routing over 6LowPAN), qui comporte un adressage hiérarchique. Ces différents protocoles proviennent de propositions de l'IETF et donc de la normalisation des réseaux ad-hoc.

Une autre caractéristique importante des protocoles des réseaux de capteurs concerne la découverte de service, qui doit permettre la mise en marche du réseau de façon automatique. L'IETF joue également un rôle important dans ce domaine, en proposant plusieurs solutions, dont une orientée vers les capteurs : LowPAN Neighbor Discovery Extension. Ce protocole est une réduction de la norme Neighbor Discovery concernant tous les éléments consommateurs d'énergie, comme les broadcast et la gestion des multicast.

La problématique principale est ici encore la sauvegarde de l'énergie lors de l'exécution des fonctions du capteur. En particulier, la partie réseau doit mener à des communications dépensant très peu d'énergie. L'Université de Berkeley a ainsi conçu un système d'exploitation et des protocoles spécifiques nommés TinyOS et Tiny protocol. Le TinyOS a été écrit dans un langage C simplifié appelé nesC, qui

est une sorte de dialecte destiné à optimiser l'utilisation de la mémoire.

4 La rfid

La RFID (Radio-Frequency Identification) a été introduite pour réaliser une identification des objets, d'où son autre nom d'étiquette électronique. Les étiquettes électroniques sont interrogés par un lecteur, qui permet de récupérer l'information d'identification. Les étiquettes électroniques sont utilisées dans de nombreuses applications, allant du suivi d'animaux à des étiquettes de produits pour magasin.

Il existe deux grands types d'étiquettes électroniques : les étiquettes passives et les étiquettes actives. Les étiquettes passives ne possèdent aucune ressource d'énergie. Elles sont allumées par un lecteur qui procure un champ électromagnétique suffisant pour générer un courant électrique permettant l'émission par une onde radio des éléments binaires stockés dans une mémoire EEPROM constituant l'identification RFID. Les étiquettes électroniques actives disposent d'une source d'alimentation électrique dans le composant. Le premier avantage très important de ces étiquettes tient à la qualité de la transmission. Une session peut être ouverte entre le lecteur et le RFID de telle sorte qu'une transmission puisse être réalisée automatiquement en cas de problème. Un autre avantage est la transmission avec une portée de plusieurs mètres entre le RFID et le lecteur, au lieu de quelques centimètres. Un inconvénient pourrait être la durée de vie de la batterie. Cependant pour une utilisation standard de quelques lectures par jour, il est possible de dépasser la dizaine d'années. Le RFID actif peut posséder une mémoire plus importante pour le stockage d'attributs associés à la valeur de l'identité.

5 Conclusion

la technologie des réseaux sans fil est une véritable révolution dans le monde de l'informatique. Pouvoir être connecté à un réseau connu ou non sans avoir à se soucier du câblage est, en effet un atout indéniable. Depuis l'an 2000, des universités déploient une structure WLAN dans leurs bâtiments. Cette solution est plus aisée et moins onéreuse à mettre en place que l'installation de prise dans chaque espace de travail.

La plupart des technologies de communication sans fils permettent de connecter un terminal mobile à un réseau d'infrastructure, en déployant des stations de base ou des points d'accès capables de jouer le rôle de passerelles entre le monde sans fils et le monde filaire. Cependant des standards tels que Wi-Fi, Bluetooth, et Zigbee permettent également de déployer des réseaux dits ad hoc, c'est-à-dire des réseaux dans lesquels des terminaux mobiles situés à portée radio les uns des autres peuvent interagir directement et spontanément, sans avoir à dépendre pour ce faire d'un quelconque élément d'infrastructure. Un réseau ad hoc est ainsi un réseau qui peut apparaître et évoluer spontanément au gré de l'apparition, de la disparition, et des déplacements des terminaux qui le constituent. En effet les réseaux sans fil apportent des débits élevés qui permettent à un PC ou à un PDA de se connecter sans se soucier du câblage et même de se déplacer lentement, à condition de ne pas sortir

de la cellule.

La diffusion massive de stations de travail de poche, telles que Pocket PC, d'une puissance comparable aux PC de bureau, va démultiplier le développement de ces réseaux sans fil, qui se présenteront comme l'entrée du réseau Internet. On donne parfois à un tel réseau, auquel on peut accéder de partout, à tout moment et à haut débit, le nom d'Internet ambiant. On peut donc s'attendre à une diversification des stations terminales de poche capables de se connecter à des réseaux Internet ambiants disponible dans tous les lieux de passage fréquentés, comme le cœur des villes, les gares, les aéroports, le métro, etc.

Chapitre 3

Partitionnement dans les réseaux mobiles ad hoc

1 Généralités

A côté des technologies *classiques* des réseaux,une autre voie a été explorée depuis les années 1970 : les réseaux mobiles ad hoc. Cela part d'un constat très simple : à l'avenir, les communications personnelles et l'informatique mobile nécessiteront une infrastructure réseau sans fil qui est facile à déployer, si possible multi-saut et capable de supporter des services multimédias. Les réseaux de téléphonie classiques et de communication mobile actuels reposent sur une base filaire fixe du réseau qu'on nomme généralement la *dorsale* (backbone en anglais). L'objectif des réseaux ad hoc est de rendre possible des communications sans fil fiables sans utiliser de dorsale filaire lourde à installer.

Tous les terminaux ou nœuds de ce type de réseau peuvent communiquer soit par un lien direct (s'ils sont suffisamment proches les uns des autres) soit grâce à la coopération d'un ou plusieurs autres membres du réseau. La difficulté dans ce type de réseau est que tout terminal agira à la fois en tant que client et en tant que serveur selon la situation. En conséquence, la plupart des tâches dites *faciles* sur des réseaux classiques deviennent nettement plus problématiques (diffusion, routage, etc.). Il faut tenir compte de la mobilité des terminaux et également du fait que tous n'ont qu'une autonomie énergétique limitée. Le délai pour la mise-au-point de tels réseaux peut sembler quelque peu pharaonique mais leurs applications pratiques seraient telles qu'il ne faut pas mesurer ses efforts. Cela inclut des applications militaires (opérations spéciales, scénarios de champs de bataille, etc.), catastrophes naturelles (incendies, tremblement de terre, inondation, etc), applications domestiques (installation d'un réseau domestique provisoire dans le cadre par exemple d'exposition ou d'un grand rassemblement,etc).

Une façon pratique de contourner la difficulté ci-dessus serait de pouvoir construire une structure hiérarchique par-dessus de façon à simuler une sorte de dorsale constituée de nœuds ou terminaux plus *adaptés* que d'autres. Cette démarche a déjà prouvé son efficacité dans le passé et constitue un problème bien connu et largement étudié dans les systèmes distribués classiques. Elle a notamment aidé dans la

résolution de plusieurs problèmes comme la minimisation de l'espace de stockage pour la communication d'informations (par exemple des tables de routages, etc.). Ainsi l'amélioration de la bande passante est effective ainsi que la distribution de ressources. La notion d'organisation en clusters a été utilisée pour les réseaux ad hoc depuis leur apparition. Cette organisation consiste à regrouper les nœuds proches géographiquement (cluster) et d'en élire localement un chef (clusterhead) selon un processus distribué d'élection de leader. Le chef ou clusterhead peut être en charge d'allocation d'adresses, de slots de communication et du routage dans son cluster. On parle alors de réseau hiérarchique. Cette hiérarchisation peut permettre une utilisation optimale des ressources, un routage plus efficace, mais aussi le passage à l'échelle.

Dans [11–14] une architecture complètement distribuée en *clusters* est proposée principalement dans le but d'établir un routage hiérarchique et de démontrer la capacité d'adaptation de ces réseaux aux changements de connectivité. Ce chapitre est organisé de la façon suivante : le paragraphe 2 donne un résumé du partitionnement en clusters dans le domaine des réseaux mobiles ad hoc. En paragraphe 3, nous détaillons l'algorithme distribué de partitionnement introduit par Basagni [4] qui intègre la notion de poids. Notre approche du partitionnement est donnée au paragraphe 4.

2 Préliminaires

Dans ce chapitre, nous proposons la première contribution de cette thèse. Il s'agit d'un algorithme distribué de partitionnement qui utilise une heuristique de poids à la différence des heuristiques de *plus petit identifiant* [36] et du *plus grand degré* [37]. Au passage n'oublions pas de noter le profit que la communauté MANET (Mobile Ad hoc NETworks) veut tirer du partitionnement malgré la différence d'opinions sur les heuristiques. Dans les solutions existantes, le partitionnement en occurrence basé sur les deux heuristiques précédemment citées, est effectuée en deux phases distinctes :

La phase d'initialisation des clusters et la phase de maintenance des clusters. La première phase est accomplie en choisissant certains nœuds pour qu'ils agissent comme l'ensemble dominant du processus de partitionnement. Les clusters sont alors formés *autour* des *clusterheads* (chef du cluster). Les algorithmes de partitionnement actuels diffèrent essentiellement sur la question : `Qui peut être clusterhead ?` [4, 36, 51].

La phase de maintenance est exposée de deux manières différentes selon les deux heuristiques. Dans [36] qui traite l'heuristique du plus petit identifiant, la maintenance des clusters en présence de la mobilité est décrite de la manière suivante : Chaque entité ou nœud du réseau décide localement d'actualiser ou non ses données concernant le cluster (sa propre appartenance ou non à son cluster). Tandis que dans [11, 14] la maintenance des clusters, est faite périodiquement à cause de la mobilité, en réappelant à intervalles réguliers l'algorithme de partitionnement.

3 Algorithme de Partitionnement Distribué : DCA et DMAC

3.1 Introduction de la notion de poids

Une troisième heuristique basée sur le poids a été proposée par Basagni [4] en 1999. L'idée de Basagni [4] est assez différente il s'est inspiré des deux phases de partitionnement (initialisation et maintenance) avec la perspective d'obtenir certaines propriétés précises. Plus précisément, étant donnée la nature dynamique des réseaux ad hoc, les nœuds ont besoin :

1. d'avoir au moins un voisin clusterhead (ceci permet des communications plus rapides entre toute paire de nœuds) ;
2. d'être affiliés au *meilleur* voisin clusterhead ;
3. de plus, deux clusterheads ne doivent pas être voisins directs, ce qui permet la bonne dispersion spatiale des clusterheads à travers le réseau.

Le poids tel qu'il est défini par Basagni [4] est un nombre entier strictement positif associé à chaque nœud : plus le poids d'un nœud est élevé, meilleur serait ce nœud pour le rôle de clusterhead. L'avantage principal de cette approche est qu'en représentant les poids avec des paramètres relatifs à la mobilité des nœuds, on peut choisir pour rôle de clusterhead les meilleurs nœuds possibles. Par exemple, si le poids d'un nœud est inversement proportionnel à sa vitesse de déplacement, alors les nœuds les moins mobiles seront sélectionnés comme clusterheads. Comme ces nœuds ne bougent pas ou très lentement en comparaison des autres nœuds, le partitionnement est garanti d'avoir une longue durée de vie et en conséquence le traitement associé à la maintenance des clusters en environnement mobile est minimisé. Pour plus de simplicité, l'auteur suppose qu'on dispose de l'unicité des poids dans le réseau.

Basagni commence par présenter l'algorithme dit DCA (Distributed Clustering Algorithm) qui est une généralisation des algorithmes de partitionnement présentés dans [14, 36]. La principale hypothèse effectuée pour la construction de l'algorithme DCA est que durant l'exécution de l'algorithme, la topologie du réseau ne change pas. En effet, cet algorithme n'est que dans sa première version. L'auteur fait également deux hypothèses plus inhabituelles dans ce contexte. Il suppose qu'un message envoyé par un nœud est reçu correctement en un temps fini (une étape) par tous ses voisins et que tout nœud connaît son identifiant, son poids, les identifiants et les poids de tous ses voisins.

L'algorithme est exécuté par chaque nœud individuellement de telle façon qu'un nœud u décide de son propre rôle (clusterhead ou nœud ordinaire) en fonction seulement de la décision de ses voisins avec des poids plus forts que lui. Donc initialement, seuls ces nœuds avec le poids le plus élevé du voisinage vont diffuser un message à leurs voisins directs, établissant qu'ils sont les *clusterheads*. Si aucun nœud v voisin de u avec un poids plus fort n'a envoyé de tel message (indiquant donc qu'il va se joindre à un autre *cluster* en tant que nœud ordinaire), alors u va envoyer un

message établissant son statut de *clusterhead*.

En dehors de la procédure initiale, l'algorithme est entièrement régi par les réceptions de messages : une procédure spécifique sera exécutée par un nœud en fonction de la nature du message reçu. L'algorithme utilise deux types de messages : `CH(u)` utilisé par un nœud u pour mettre au courant ses voisins qu'il sera *clusterhead* et `JOIN(v,u)`, avec lequel v communique à ses voisins qu'il va faire partie du cluster dont le *clusterhead* est u

3.2 Modélisation et Notations

Afin de formaliser notre présentation, prenons quelques conventions de notation. Le réseau peut être représenté par un graphe non orienté *G=(V,E)* avec *V*, l'ensemble des terminaux mobiles v_i et *E*, l'ensemble e_i des liens de communication sans fil. On considère un sous-cas par rapport au cas général : les liens de communication sont symétriques (si le nœud p est voisin du nœud q, alors q est aussi voisin de p). Deux œuds u et v sont en mesure de communiquer si et seulement si la distance euclidienne entre u et v notée $d(u, v)$ est inférieure ou égale à la portée de communication r considéré ici homogène. A chaque nœud v est associé un identifiant (ID_v) unique et un poids unique w_v (un entier positif). Le réseau pondéré résultant est donné à la figure 3.1.

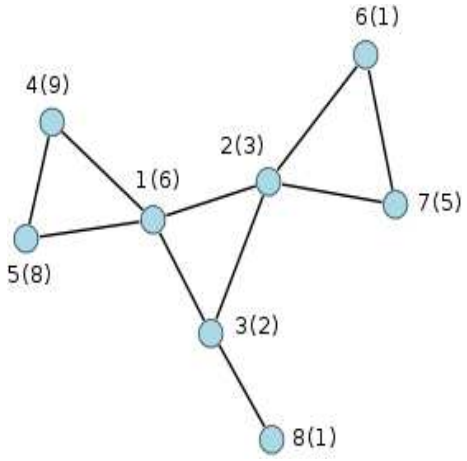


FIG. 3.1 – Réseau ad hoc pondéré.

L'ensemble des voisins d'un nœud $v \in V$ est noté par $\Gamma(v)$ tels que

$$\Gamma(v) = \bigcup_{v' \in V, v' \neq v} \{v' | d(v, v') < r\}$$

r est le rayon de transmission du nœud v. Le voisinage d'un nœud u est l'ensemble des nœuds localisés dans son rayon de transmission. Soit S l'ensemble des *clusterhead*, alors $S \subseteq V(G)$ tels que

$$\bigcup_{v \in S} \Gamma(v) = V(G).$$

En théorie des graphes S est appelé *ensemble dominant* ce qui signifie que tout sommet de G appartient à S ou à un voisin dans S. Dans l'explication de l'algorithme, on utilise les notations suivantes :

1. u, le nœud générique exécutant l'algorithme (à partir de maintenant, on dira que u est l'identifiant de ce nœud et w_u son poids) ;

2. $Cluster(u)$, l'ensemble des nœuds du *cluster* de u. Cette variable initialisée à $\oslash$ n'est actualisée que si u est un *clusterhead* ;

3. *Clusterhead*, la variable dans laquelle chaque nœud enregistre l'identifiant du *clusterhead* auquel il s'est soumis. La valeur d'initialisation est nil ;

4. $Ch(-)$ et $Join(-,-)$, deux variables booléennes. Le nœud v met $Ch(u)$, (avec $u \in \{v\} \bigcup \Gamma(v)$) a **vrai** dés lors qu'un message de type $CH(v)$ est émis avec $(v = u)$ ou quand il reçoit de u $(u \neq v, u \in \Gamma(v))$ un message de type $CH(u)$. La variable booléenne $Join(u,t), u \in \Gamma(v), t \in V$, est mise à **vrai** par v quand v reçoit un message de type $JOIN(u,t)$ de u. Ces deux variables sont initialisées à **faux**.

5. $\wedge$ et $\vee$ représentent respectivement les opérateurs booléens **ET** et **OU**.

3.3 Détails de l'algorithme de partitionnement distribué

L'algorithme de partitionnement distribué proposé par Basagni [4] comprend deux phases distinctes : la phase *d'initialisation* qui décrit l'ensemble des procédures nécessaires à la formation des *clusters* et la phase de maintenance qui introduit d'autres procédures pour le maintien des clusters face à la mobilité des nœuds.

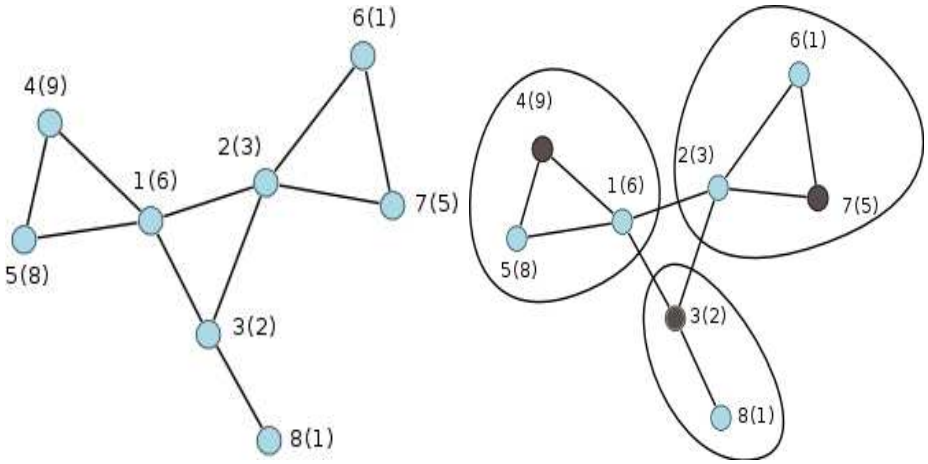
L'initialisation

Tous les nœuds du réseau démarrent l'exécution de l'algorithme en même temps, en utilisant une même procédure d'initialisation. Seuls les nœuds qui ont le poids le plus élevé par rapport à tous leurs voisins directs vont envoyer un message de type *CH(nœuds de poids forts)*. Grâce à la nature des poids (des nombres entiers), il existe toujours au moins un nœud v qui transmet le message *CH(v)*. Tous les autres nœuds se contentent d'attendre un tel message.

A partir de là, on dispose de deux procédures spécifiques dont l'exécution est déclenchée par les réceptions de messages :

• Lors de la réception d'un message CH venant d'un voisin u, le nœud v vérifie d'abord s'il a reçu de tous ses voisins z avec $z \in \Gamma(v)$, tels que $w_z > w_u$ un message $JOIN(x,z)$ avec x un nœud quelconque du réseau (ceci est en fait enregistré dans la variable booléenne correspondante). Dans le cas où, v ne recevra pas de message de type CH de l'un de ses voisins z et si u est le nœud avec le plus fort poids dans le voisinage de v alors v rejoindra u et quitte l'exécution de l'algorithme. Le nœud v sera désormais rattaché au *cluster* dont u est le *clusterhead*. S'il reste toujours au moins un nœud z tel que $w_z > w_u$, qui n'a pas encore envoyé de message, alors le nœud v se contente d'enregistrer dans la variable $Ch(u)$ que u lui a envoyé un message de type CH, et se contente d'attendre un message de z.

• Lors de la réception d'un message $JOIN(u,t)$, le nœud v vérifie s'il a précédemment envoyé un message CH (c'est à dire s'il a décidé d'être un *clusterhead*). Si c'est bien le cas, v vérifie si le nœud u veut rejoindre le *cluster* de v (c'est à dire $v = t$) et actualise, si besoin son ensemble *cluster(v)*. Ainsi si tous les voisins z de v, tels que $w_z < w_v$ ont communiqué leur volonté de rejoindre un *cluster*, v quitte l'exécution du DCA. Notez bien dans ce cas que le nœud v ne prête pas la moindre attention à ses voisins (s'il y en a) tel que $w_y > w_v$, car ces nœuds ont assurément rejoint un nœud x de poids supérieur à celui de v. Cela permet à v d'être $Clusterhead$ bien que certains de ses voisins aient un poids plus élevé que le sien. Si le nœud v n'a pas envoyé un message de type CH, avant de décider quel sera son rôle, il a besoin de savoir ce que tous les nœuds z, tels que $w_z > w_v$, ont décidé pour eux-mêmes. Si v a reçu un message de tous les nœuds de ce type, alors il vérifie la nature des messages reçus. Si ce sont tous des messages de type $JOIN$, cela signifie que tous ces voisins z ont décidé de rejoindre un $Cluster$ en tant que nœud *ordinaire*. Dans ce cas v, sera $Clusterhead$ en exécutant les opérations nécessaires (c'est à dire il envoie un message de type CH, il actualise son ensemble $Cluster(v)$, et met la variable $Ch(v)$ à *vrai* et la variable $Clusterhead$ à v. A ce moment, v vérifie aussi si chacun de ces voisins y tel que $w_y < w_v$ a déjà rejoint un autre $Cluster$. Si c'est le cas v, quitte l'exécution de l'algorithme : il sera $Clusterhead$ d'un $Cluster$ composé de lui-même uniquement. Sinon, si v a reçu au moins un message de type CH, alors il rejoint le $Cluster$ du voisin au poids le plus élevé qui lui a envoyé précédemment un message CH. Puis il quitte l'exécution du DCA. Notez bien qu'un nœud quitte toujours l'exécution de l'algorithme une fois qu'il a envoyé un message $JOIN$. Un exemple d'exécution du DCA est donné à la figure 3.2.



(a) Un Réseau ad hoc G pondéré de poids w, $1 \leq w \leq 9$

(b) Partitionnement de G

FIG. 3.2 – Partitionnement distribué

La maintenance des clusters

L'algorithme de Basagni exposé ci-dessus convient pour des réseaux *quasi statiques* (c'est à dire son algorithme est approprié pour installer une architecture en

clusters dans un réseau dont les nœuds ne bougent pas beaucoup : mobilité lente). En suivant le modèle présenté par Basagni [4] , l'utilisation du DCA en présence de mobilité peut être décrite comme une succession de partitionnement périodique (c'est à dire les *clusters* sont continuellement reconstruits de façon à s'adapter aux variations imprévisibles de la topologie). Pour éviter un partitionnement périodique du réseau Basagni a proposé une autre version du DCA appelée DMAC (Distributed and Mobility-Adaptive Clustering) qui s'adapte à la mobilité.

La première différence fondamentale entre le *DMAC*et le *DCA* est que le *DMAC* va gérer tant l'initialisation que la maintenance de l'organisation des *clusters* malgré la mobilité des nœuds tout en assurant les trois propriétés évoquées précédemment. L'autre différence est que désormais on n'a plus besoin de supposer que la topologie du réseau est statique durant la construction des *clusters*. L'adaptation aux changements de la topologie du réseau est rendue possible en laissant chaque nœud réagir non seulement à la réception d'un message venant d'autres nœuds, mais également à la *défaillance d'un lien* de communication vers un autre nœud (probablement causée par la défaillance du nœud ou par les mouvements des terminaux ou nœuds) ou la présence d'un *nouveau lien*.

Basagni [4] a donc présenté deux algorithmes, le *DCA* et le *DMAC*, de partitionnement des nœuds de tout réseau mobile *ad hoc* en un ensemble de *clusters* composés chacun d'un *clusterhead* et éventuellement de plusieurs nœuds ordinaires. Un nouveau critère de sélection du *clusterhead* basé sur les poids a été introduit, dans le seul but de prendre en compte la notion de mobilité, ce qui n'était pas le cas des précédents algorithmes de partitionnement. La mobilité n'est pas le seul facteur dont il faudra tenir compte quand on parle de réseaux mobiles ad hoc. En plus, dans son algorithme aucune proposition ou hypothèse n'est faite sur la taille des *clusters*. Plus un cluster est grand, plus l'énergie dépensée par le *clusterhead* est élevée. Tout algorithme destiné aux réseaux mobiles de type ad hoc doit prendre en compte au moins la mobilité, la bande passante, la consommation d'énergie des nœuds et leur puissance de transmission. Dans le paragraphe qui suit, nous proposons notre algorithme de partitionnement en *clusters* basé sur la notion de poids (non nécessairement unique) et qui prend en compte les contraintes telles que : la mobilité, la consommation d'énergie et la taille des clusters.

4 Partitionnement Sans Unicité du Poids : PSUP

Nous proposons ici un nouvel algorithme de partitionnement basé sur la notion de poids (non nécessairement unique) et qui prend en compte certains paramètres du réseau. On note par P_v le poids d'un nœud v. On intègre dans le poids de chaque nœud les paramètres suivants :

• *La mobilité :* avant qu'un nœud v ne s'attribue le statut de *clusterhead* il doit d'abord mesurer sa mobilité noté ω_v (donnée par l'inverse de sa vitesse moyenne de déplacement). Si cette mobilité est plus élevée (vitesse moyenne moins élevée) que celles de ses voisins, le nœud peut dans ce cas prétendre le rôle de *clusterhead*. Ce choix permet de réduire le nombre d'appels à la procédure de partitionnement mais aussi de conserver plus longtemps une structure de clusters. Les nœuds les plus

mobiles ne peuvent pas prétendre au rôle de *clusterhead*.

• *L'énergie résiduelle :* la gestion des ressources et du routage dans un cluster font que le clusterhead dépense plus d'énergie qu'un nœud ordinaire. De ce fait, il serait judicieux d'octroyer le rôle de *clusterhead* aux nœuds qui disposent d'assez d'énergie (supérieure à un seuil donné qu'on note par E_s). Lorsqu'un clusterhead ne dispose plus d'assez d'énergie l'algorithme de partitionnement est réexécuté de nouveau afin de reconstituer une structure en clusters plus stable. L'énergie résiduelle est propre à chaque nœud v, elle est noté par $E_r(v)$. Elle n'a pas d'impact lors de la première exécution de l'algorithme de partitionnement puisque à ce stade les nœuds disposent tous de la même quantité d'énergie.

• *La différence de degré :* celui-ci permet de limiter (borner) la taille d'un cluster autour d'une valeur de seuil δ donnée, autrement dit on choisit comme clusterhead les nœuds ayant la plus grande différence de degré. Pour un nœud v nous notons cette différence de degré par ϕ_v. Cette limitation permet au *clusterhead* une meilleure gestion du contrôle d'accès au médium (MAC). Nous avons introduit ce paramètre dans le but de réduire la charge de chaque clusterhead ce qui permet d'allonger la durée de vie de leurs batteries. Plus un cluster est dense plus l'énergie consommée est importante.

• *L'identifiant :* ce dernier paramètre identifie chaque nœud. Il est pris en compte lorsqu'une égalité apparaît sur les paramètres cités ci-dessus. Si ce paramètre doit être utilisé pour départager les nœuds sur le statut de clusterhead, on choisira celui qui a le plus grand identifiant.

4.1 Définition du modèle

On suppose un réseau ad hoc de n nœuds déployés aléatoirement dans une grille X de taille $|X|$. Le réseau peut être représenté par un graphe non orienté *G=(V,E)* avec *V*, l'ensemble des terminaux mobiles v_i et *E*, l'ensemble e_i des liens de communication sans fil. On considère un sous-cas par rapport au cas général : les liens de communication sont bidirectionnels (si le nœud u peut communiquer avec le nœud v, alors v est aussi en mesure de communiquer avec le nœud u). Les nœuds sont homogènes disposant chacun d'un même rayon de transmission r. Le voisinage d'un nœud v noté $\Gamma(v)$ tel que

$$\Gamma(v) = \bigcup_{v' \in V, v' \neq v} \{v' | d(v, v') < r\} \tag{3.1}$$

On note par d_v son degré défini par

$$d_v = |\Gamma(v)| = \sum_{v' \in V, v' \neq v} \{v' | d(v, v') < r\} \tag{3.2}$$

La *différence de degré* du nœud v, est donnée par

$$\phi_v = \delta - d_v \tag{3.3}$$

avec δ une constante donnée.

Nous avons introduit cette constante δ dans le but d'élire comme *clusterhead* les nœuds les moins denses en termes de voisins. Pour éviter que les nœuds se déplacent n'importe comment dans la grille, nous avons défini un autre paramètre, d_{max}, la distance maximale autorisée lors d'un déplacement. Quand un nœud v désire se déplacer il tire au hasard une valeur Δ ($0 < \Delta \leq d_{max}$). Pour éviter que le déplacement sur une distance Δ ne mène un nœud v en dehors de la grille alors v se déplace d'une distance η vers le bord puis d'une distance $\Delta - \eta$ vers le centre. La distance de déplacement Δ est parcourue entre deux instants t_1 et t_2, et on évalue la vitesse à l'instant t_2 par $v_2 = \frac{\Delta}{t_2 - t_1}$. A chaque déplacement le nœud évalue sa vitesse entre l'instant de départ et l'instant d'arrivée. La vitesse moyenne d'un nœud v est dans ce cas mesurée par

$$M_v = \frac{1}{m} \sum_{i=1}^{m} v_i. \quad m : nombre\ de\ dplacement \tag{3.4}$$

Nous mesurons la mobilité d'un nœud v par

$$\omega_v = 1/M_v \tag{3.5}$$

On suppose qu'un nœud est capable d'évaluer son énergie résiduelle $E_r(v)$. Le poids d'un nœud v sera un quadruplet de valeurs exprimé de la manière suivante :

$$P_v = \{\omega_v, E_r(v), \phi_v, ID_v\} \tag{3.6}$$

4.2 Identification partielle des nœuds

Les différents nœuds composant le réseau sont déployés aléatoirement dans la région X de taille $|X|$. Le réseau est supposé connexe après le déploiement, mais les nœuds demeurent indistinguables les uns des autres. Si n le nombre total des nœuds et N une borne supérieure de n. L'identification partielle permet d'attribuer un identifiant temporaire à chaque nœud selon une loi uniforme d'intervalle $[1 \ldots N^3]$. Ceci nous permet d'annoncer le théorème ci-dessous basé sur la technique de distribution uniforme appelée `Occupancy Problem` [17] :

Théorème 3.1 *Soit n le nombre total des nœuds connus par tous les terminaux et N une borne supérieure de n alors la probabilité d'octroyer un identifiant unique à chaque nœud selon une loi uniforme d'intervalle $[1 \ldots N^3]$ est supérieure à* $\exp(\Theta(n^2/N^3))$.

Preuve 3.1 *La preuve est un résultat d'une application qui consiste à placer r billes dans m cases. Notant par φ l'univers des solutions possibles pour cette expérience,ψ l'ensemble des solutions sans répétition et p la probabilité d'obtenir une bille par case.*

$$p = \frac{Card(\psi)}{Card(\varphi)} = \frac{m(m-1)(m-2)\dots m-r+1)}{m^r}$$
$$= (1 - \frac{1}{m})(1 - \frac{2}{m}) \dots (1 - \frac{(r-1)}{m}$$
$$= (1 - \frac{2}{m} - \frac{1}{m} + \frac{1}{m^2}) \dots (1 - \frac{(r-1)}{m}) \approx 1 - \frac{(1+2+\dots r-1)}{m}$$
Si r est petit alors $p = 1 - \frac{r(r-1)}{2m}$

Si r est grand alors $p = \log(p) = \log(1 - \frac{r(r-1)}{2m})$
$$\approx -\frac{r(r-1)}{2m} \approx \exp(-\frac{r(r-1)}{2m})$$
$$> \exp(-\Theta(n^2/N^3))$$
en posant $r = n$ *et* $m = N^3$

4.3 Découverte des voisins à un saut

Une fois qu'une identité est obtenue, chaque nœud v doit procéder à la découverte de ses voisins à un saut (voisins immédiats). La seule information disponible jusque là est la taille n du réseau par conséquent l'algorithme de découverte doit être exécuté par chaque nœud avant le début de notre algorithme de partitionnement. Notre approche est similaire à celle de [50]. Il s'agit d'un modèle probabiliste dans lequel on a supposé que N reste le degré maximal pour chaque nœud et en même temps la borne supérieure de n (avec n le nombre total de nœuds). On suppose qu'un message M[1] envoyé par un nœud v à un voisin u est toujours reçu correctement en un temps fini et le délai de propagation est constant. L'algorithme qui en résulte pour un nœud générique v est donné ci-dessous.

Algorithm 1 Découverte(N)

1: **begin**
2: $\Gamma(v) := \oslash$
3: $\Gamma(v) := \Gamma(v) \bigcup \{ID_v\}$
4: $M := ID_v$
5: **for** *k allant 1 a N* **do**
6: *Envoyer M avec la probabilit* $1/d_v^k$
7: **end for**
8: **for** *Chaque M reu* **do**
9: $\Gamma(v) := \Gamma(v) \bigcup M.ID$
10: **end for**
11: **end**

A la fin de cet algorithme chaque nœud disposera de la liste complète de ses voisins à un saut. Le partitionnement en clusters comprend formellement deux

[1] un cours message ne contenant que l'identité du nœud

étapes : une étape de construction des clusters en donnant le statut de *clusterhead* aux nœuds les mieux adaptés selon les critères choisis. Une seconde étape consiste à la maintenance de ces clusters face aux changements liés à la topologie. Notre proposition va gérer tant la construction que la maintenance des clusters face à la mobilité et aux variations des k paramètres contenus dans l'expression du poids.

4.4 Détails de notre algorithme de partitionnement

L'algorithme est exécuté par chaque nœud individuellement de telle façon qu'un nœud v décide de son propre rôle (clusterhead ou nœud ordinaire) en fonction seulement de la décision de ses voisins avec des poids plus forts que lui. Le poids n'est plus un simple entier positif comme dans [4], mais un ensemble de valeurs réelles qui reflètent l'état du réseau. Les trois premières valeurs expriment les informations relatives au nœud et le 4^{ieme} paramètre permet son identification. On considère l'hypothèse supplémentaire suivante : tout nœud est mis au courant de la défaillance d'un lien ou de la présence d'un nouveau lien de communication par un service réseau de bas niveau qui déclenchera ainsi une procédure adaptée. Un nœud u peut prétendre le rôle de clusterhead si son poids est plus grand le poids de tous ces voisins. Le poids P_v d'un nœud v est plus grand que celui du nœud u si et seulement si l'une des trois conditions est vérifiée :

1. $(\omega_u > \omega_z) \wedge (E_r(u) > E_r(z)) \wedge (\phi(u) > \phi(z))$
2. $(\omega_u = \omega_z) \wedge (E_r(u) = E_r(z)) \wedge (\phi(u) = \phi(z)) \wedge (ID_u > ID_z)$
3. $(\omega_u \geq \omega_z)$
   $\wedge$
   $(E_r(u) \geq E_r(z)) \vee (E_r(u) \geq E_s)$
   $\wedge$
   $(\phi(u) \geq \phi(z)) \vee \{(E_r(u) \geq E_s) \vee (E_r(u) \geq E_r(z))\}$

La troisième condition peut se résumer de la manière suivante : il s'agit de donner un ordre d'importance ou de priorité aux paramètres qui constituent le poids. Cette priorité est évaluée de gauche à droite, plus un paramètre est à gauche plus il est prioritaire. En sens lorsqu'un nœud u domine un voisin v sur la mobilité $(w_u > w_v)$ alors le poids de u est dit plus grand que le poids de v si et seulement si $E_r(u) \geq E_s$, quelque soit le résultat de comparaison sur les paramètres de droite et de même rang.

Par exemple lorsque le poids de v est $(2.4, 3.1, 39.3, 2)$ et que celui de u est $(2.4, 2.5, 38.68, 4)$ on dira que v remporte sur u car le second paramètre donne l'avantage au nœud v. Lorsque le poids de v est $(2.4, 1, 38, 2)$ et celui de u $(2.3, 5, 39.68, 4)$ on dira également que v remporte sur u car le premier paramètre donne l'avantage au nœud v.

Lorsque $P = (0, 0, 0, 0, ID)$ nous retrouvons la première approche de *Gerla et Tsaï* [36] basée sur l'heuristique du plus petit identifiant, un nœud générique v est élu *clusterhead* lorsque son poids est plus fort que ceux de ses voisins. Nous retrouvons la seconde approche basée sur l'heuristique du plus grand degré de Gerla et Tsaï [37] en posant $\delta = 0$ et en prenant pour chaque nœud v la valeur absolue de sa

différence de degré ϕ_v, $P = (0, 0, 0, |\phi_v|, 0)$.

Ce qui suit est la description complète des procédures composant notre algorithme, exécutées par chaque nœud v.

◦ `Construction de la structure` : Lors de l'installation de la structure de *cluster* ou quand un nœud v est ajouté au réseau ou encore quand la mobilité ou l'énergie d'un nœud est mise en cause, tout nœud exécute la procédure de construction de façon à (re)déterminer son propre rôle. Si parmi ses voisins, il y a au moins un *clusterhead* avec un poids plus élevé que le sien, alors v le rejoint. Autrement, il sera un *clusterhead*. Notez bien qu'un voisin avec un plus fort poids qui n'a pas encore décidé de son propre rôle (cela peut arriver lors de la première construction du réseau, ou quand deux nœuds ou plus sont ajoutés simultanément au réseau), peut éventuellement envoyer un message par la suite (tout nœud finit forcément par exécuter cette procédure de construction). Si ce message est du type CH alors v renonce à son rang de *clusterhead* et se soumet à ce nouveau *clusterhead*.

Algorithm 2 `Procedure Construct`

begin
$\Gamma(v) := \bigcup_{v' \in V, v' \neq v} \{v' | d(v, v') < r\}$
$d_v := |\Gamma(v)| = \sum_{v' \in V, v' \neq v} \{v' | d(v, v') < r\}$
$\phi_v := \delta - d_v$
$M_v := \frac{1}{m} \sum_{i=1}^{m} v_i$
$\omega_v := 1/M_v$
$P := (\omega_v, E_r, \phi_v, d_v, ID_v)$
Envoyer P_v
if $(\exists z \in \Gamma(v)/P_z > P_v) \wedge (Ch(z) \neq faux)$ **then**
    $x := max_{P_z > P_v}\{z/Ch(z) = vrai\}$
    Envoyer $JOIN(v, x)$
    $Clusterhead := x$
else
    Envoyer $CH(v)$
    $Ch(v) :=$ **vrai**
    $Clusterhead := v$
    $Cluster(v) := \{v\}$
end if
end

◦ `Lors de la réception d'un message` $CH(u)$: Quand un voisin u de v devient un *clusterhead*, il envoie à l'ensemble de ses voisins directs, notamment v, un message $CH(u)$. Lors de la réception de ce message, le nœud v vérifie s'il va être obligé de se soumettre à u (c'est à dire il vérifie si le poids de u est plus élevé que celui du *clusterhead* auquel il est soumis). Dans ce cas, indépendamment de son propre rôle actuel, v rejoint le *cluster* de u ;

Algorithm 3 `Procedure réception_CH(u)`

```
begin
if P_u > P_clusterhead then
    Envoyer JOIN(v, u)
    Clusterhead := u
    if Ch(v) then
        Ch(v) := faux
    end if
end if
end
```

∘ `Lors de la réception d'un message` $JOIN(u, z)$: Le comportement du nœud v dépend du fait qu'il est actuellement *clusterhead* ou non. Si c'est le cas, v doit vérifier si u rejoint son *clusterhead* ($z = v$: dans ce cas, u est ajouté à $Cluster(v)$) ou si u appartenait à son *cluster* et rejoint maintenant un autre *cluster*($z \neq v$: dans ce cas, u est enlevé de $Cluster(v)$). Si v n'est pas un *clusterhead*, il doit vérifier si u était son *clusterhead*. Uniquement si c'est le cas, v doit décider quel sera son rôle : il rejoindra le *clusterhead* x de plus fort poids tel que le poids de x est strictement supérieur à son propre poids, si un tel nœud existe. Autrement, il sera son propre *clusterhead*.

Algorithm 4 `Procedure réception_JOIN(u,z)`

```
begin
if (Ch(v)) then
    if z = v then
        Cluster(v) := Cluster(v) ∪ {u}
    else
        if u ∈ Cluster(v) then
            Cluster(v) := Cluster(v) \ {u}
        end if
    end if
else
    if Clusterhead = u then
        if {z ∈ Γ(v)/P_z > P_v ∧ Ch(z) = vrai} then
            x := max_{P_z > P_v}{z/Ch(z)}
            Envoyer JOIN(v, x)
            Clusterhead := x
        end if
    end if
else
    Envoyer CH(v)
    Ch(v) := vrai
    Clusterhead := v
    Cluster(v) := {v}
end if
end
```

∘ `Défaillance d'un lien de communication` : Quel que soit le moment où un nœud v est mis au courant de la défaillance d'un lien de communication vers un de ses voisins directs u, le nœud v commence par vérifier si son propre rôle est actuellement celui de *clusterhead* et si u appartenait ou non à son *cluster*. Si c'est le cas, v enlève u de *Cluster(v)*. Si v est un nœud ordinaire et que u était son *clusterhead*, v se retrouve orphelin et doit déterminer quel sera son nouveau rôle dans l'avenir. Dans ce but, v vérifie s'il existe au moins un *clusterhead* z dans son voisinage qui a un poids plus fort que le sien. Si c'est le cas, alors v rejoint le *clusterhead* de plus fort poids. Sinon, il devient son propre *clusterhead*. . ∘ `Apparition d'un nouveau`

Algorithm 5 `Procedure defaillance_lien(u)`

```
begin
if (Ch(v) ∧ (u ∈ Cluster(v)) then
    Cluster(v) := Cluster(v)\{u}
else
    if Clusterhead = v then
        if {z ∈ Γ(v)/P_z > P_v ∧ Ch(z) ≠ ∅} then
            x := max_{P_z>P_v}{z/Ch(z)}
            Envoyer JOIN(v, x)
            Clusterhead := x
        end if
    end if
else
    Envoyer CH(v)
    Ch(v) := vrai
    Clusterhead := v
    Cluster(v) := {v}
end if
end
```

`lien de communication` : Quand v est mis au courant de la présence d'un nouveau voisin u, il vérifie si u est un *clusterhead*. Si c'est le cas et si u a un poids plus fort que celui du *clusterhead* actuel de v alors indépendamment de son propre rôle actuel, v se soumet à u ;

Algorithm 6 `Procedure nouveau_lien(u)`

```
begin
if (CH(u) then
    if P_u > P_clusterhead then
        Envoyer JOIN(v, u)
        Clusterhead := u
        if CH(v) then
            CH(v) := faux
        end if
    end if
end if
end
```

On conclut finalement que les procédures décrites ci-dessus permettent d'obtenir et de maintenir pour tout réseau *ad hoc* un partitionnement qui satisfait les trois propriétés énoncées précédemment.

5 Simulations

5.1 Environnement de simulations

Nous présentons dans cette section les résultats de simulations obtenues. Ces résultats montrent l'influence que peut avoir l'heuristique sur le partitionnement. Nous avons avons comparé les trois heuristiques sur le nombre de messages échangés. Nous avons utilisé deux simulateurs étroitement liés OMNET++-3.3 et Mobility-fw.0p2. Les nœuds sont placés dans une grille de taille 100x100 selon une distribution uniforme que propose le simulateur, avec une faible mobilité ($10m/s$ au plus). La vitesse de déplacement n'est pas nécessairement unique chaque nœud peut se déplacer à une vitesse aléatoire choisie selon un générateur de nombre aléatoire Mersenne Twister (intégré au simulateur OMNET++).

Le simulateur utilise le protocole CSMA, les messages corrompus ne sont pas pris en compte. Tous les messages émis durant le processus de simulation sont délivrés correctement. La diffusion par un nœud u vers n nœuds est gérée comme n transmissions en *unicast*. Les nœuds sont homogènes de rayon de transmission égale à 13. Les nœuds se déplacent selon des directions aléatoires sans effectuer de pause entre le point de départ et le point d'arrivée. Nous avons simuler 100 $fois$ chaque scénario avec 34 nœuds au total.

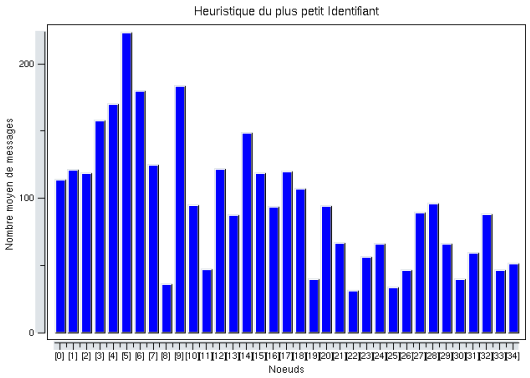
5.2 Résultats de simulations

Les résultats ci-dessous nous ont permis de comparer les heuristiques de partitionnement existants d'une part et d'autre part de les comparer à notre heuristique de partitionnement basée sur la notion de poids. Selon l'ordre d'apparition des paramètres qui composent le poids on s'est rendu compte que les résultats obtenus sur le nombre moyen de clusters ne sont pas identiques quelque soit le type de mobilité.

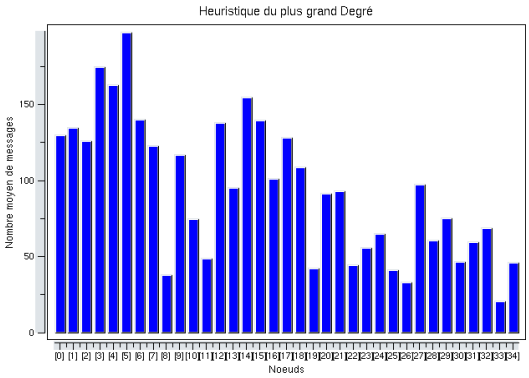
Comparaison selon le nombre moyen de messages

Les algorithmes de partitionnement sont entièrement régis par réception de messages ; en effet les comparer en terme de messages constitue un moyen permettant d'évaluer la charge du réseau. Les différentes figures ci-dessous nous permet de comparer notre heuristique basée sur le poids aux heuristiques du plus petit identifiant et du plus grand degré. La comparaison est faite sur le nombre moyen de messages (entrant et sortant) par nœud sur 100 expériences avec une légère variation de la topologie. Les figures ci-dessous évaluent le coût des heuristiques en termes de messages échangés tout au long du processus de partitionnement.

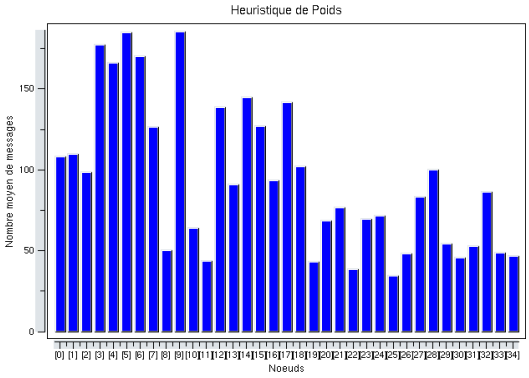
.



(a) Heuristique du plus petit identifiant



(b) Heuristique du plus grand degré



(c) Heuristique basée sur le poids

FIG. 3.3 – Nombre moyen de messages

La figure 3.4 donne un aperçu du nombre moyen de messages (émission et réception confondues) en fonction du temps d'exécution.

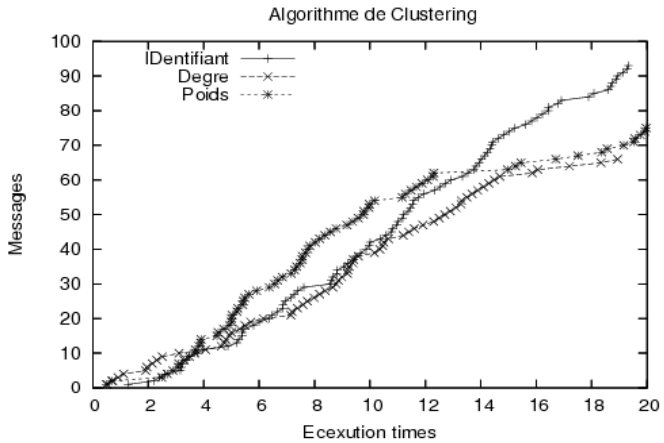


FIG. 3.4 – Messages vs temps

La figure 3.5 donne un aperçu du nombre moyen de messages (émission et réception confondues) sur différentes topologies en posant $\delta = 4$.

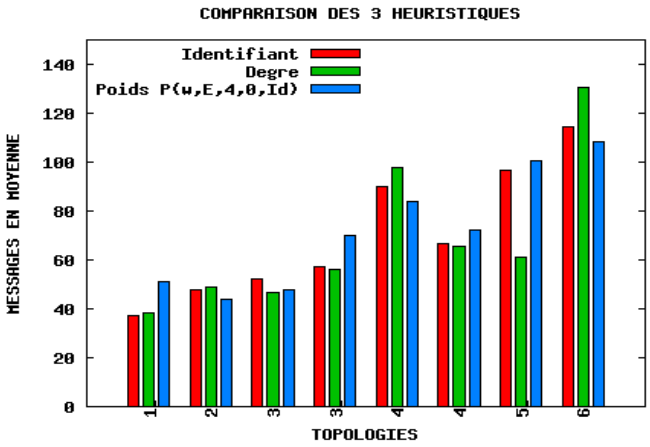


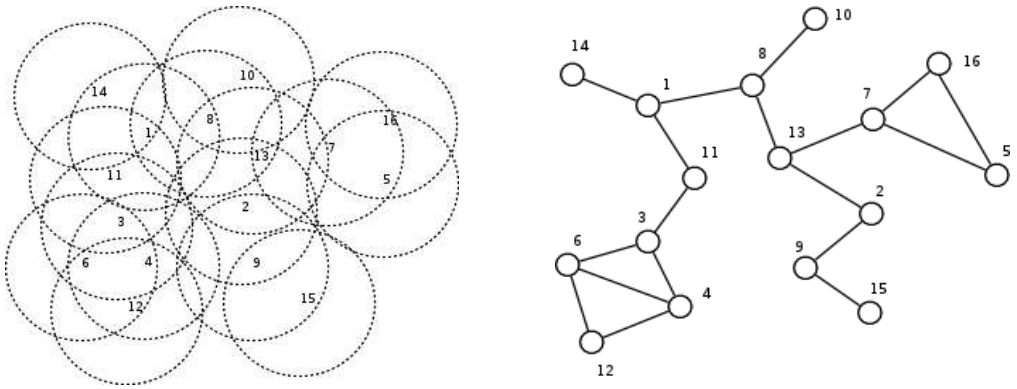
FIG. 3.5 – Messages vs topologies variables

Statistiques de notre heuristique de poids

Nous démontrons avec quelques figures et résultats les différents processus de notre algorithme de partitionnement ($PSUP$), les valeurs numériques ici sont obtenues en exécutant $PSUP$ sur 16 nœuds comme il est montré à la figure 3.7 et à la table 3.1. La figure 3.6.(a) montre la configuration initiale où chaque nœud ne possède comme information que son identifiant id, les cercles en pointillés représentent la portée radio de chaque nœud. La diffusion permet à chaque nœud de découvrir l'ensemble de ses voisins ce qui conduit à la figure 3.6.(b). Une arête entre deux nœuds u et v signifie que u et v sont voisins (liaison bidirectionnelle).

La figure 3.8(a) simule une configuration de mobilité ; une *courte flèche* modélise un nœud avec une faible mobilité et une *longue flèche* modélise un nœud avec une haute mobilité. Les paramètres qui composent le poids P sont ici l'inverse de la vitesse moyenne ω, l'énergie résiduelle E_r, la différence de degré ϕ_v et enfin l'identifiant ID_v, le poids d'un nœud v sera alors $P_v = (\omega_v, E_r, \phi_v, ID_v)$. Pour l'illustration de la méthode on a choisi des valeurs arbitraires pour l'inverse de la vitesse moyenne et l'énergie résiduelle. La différence de degré est mesurée par rapport à une constante seuil δ fixée ici à 2 ($\delta = 2$) ce qui veut dire qu'un nœud v ne peut prétendre le rôle de *clusterhead* que lorsque son voisinage compte plus ou moins δ voisins. L'autre constante seuil concerne l'énergie résiduelle $E_s = 10\%$, ce qui signifie que le rôle de *clusterhead* ne sera accordé qu'aux nœuds possédant un niveau d'énergie supérieur à 10%.

La table 3.1 résume l'exécution de l'algorithme vis à vis de ces paramètres.



(a) Configuration initiale (b) Voisins identifiés

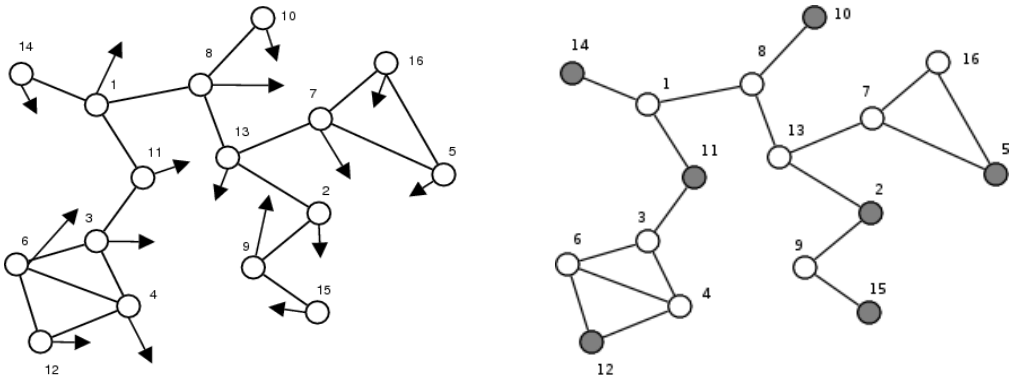
FIG. 3.6 – Configuration et Identification

Après l'identification chaque nœud est libre de se déplacer dans la région d'intérêt, on suppose que ce déplacement n'affecte pas la connectivité du réseau. A cette étape du partitionnemet chaque nœud v doit transmettre son poids à ses voisins immédiats comme il a été explicité dans la procédure de construction des clusters (algorithme2).A la fin de la procédure on obtient un réseau partitionné à la figure 3.7.(b) où les *clusterheads* sont en gras et les nœuds ordinaires en clair. Deux *clusterheads* ne sont pas voisins et chaque *clusterhead* a au plus deux voisins, une valeur qui se rapproche de notre idée de base qui consistait à sélectionner des

ω_v	E_r	ϕ_v	d_v	ID_v	$Statut = clusterhead$
0.019	45.8	-1	3	1	non
0.045	19.25	0	2	2	oui
0.056	24.33	-1	3	3	non
0.054	23.8	-1	3	4	non
0.077	18.5	0	2	5	oui
0.011	20.6	-1	3	6	non
0.027	11.5	-1	3	7	non
0.034	14.74	-1	3	8	non
0.012	17.5	0	2	9	non
0.055	23.25	1	1	10	oui
0.024	16.05	0	2	11	oui
0.015	30.73	0	2	12	oui
0.004	18.00	-1	3	13	non
0.071	12.52	1	1	14	oui
0.068	21.77	1	1	15	oui
0.034	16.82	0	2	16	non

TAB. 3.1 – Exécution du *PSUP*

clusterheads moins denses. La connectivité du réseau peut être réalisée lorsque l'on s'appuie sur l'ensemble des clusters comme *dorsale* en octroyant aux clusterheads une puissance de transmission plus élevée.



(a) Déplacement des nœuds (b) Partitionnement en clusters

FIG. 3.7 – Déplacement et Partitionnement

Les figures 3.8 et 3.9 comparent notre heuristique de poids aux heuristiques classiques de partitionnement selon deux type de mobilité. Lorsque la mobilité est basse le nombre moyen de clusters varie légèrement d'une heuristique à l'autre. Toutefois l'heuristique basée sur l'identifiant enregistre plus de mise à jour que les autres. Mais lorsque le mobilité est forte les changements de statut (clusterhead ou nœud ordinaire) est considérable quelle que soit l'heuristique. L'ordre d'apparition des paramètres dans notre système de poids a pour effet de mesurer la sensibilité de l'algorithme. On peut dans ce cas introduire une notion de priorité de paramètres allant de la gauche à la droite. Plus un paramètre est à gauche plus il est prioritaire.

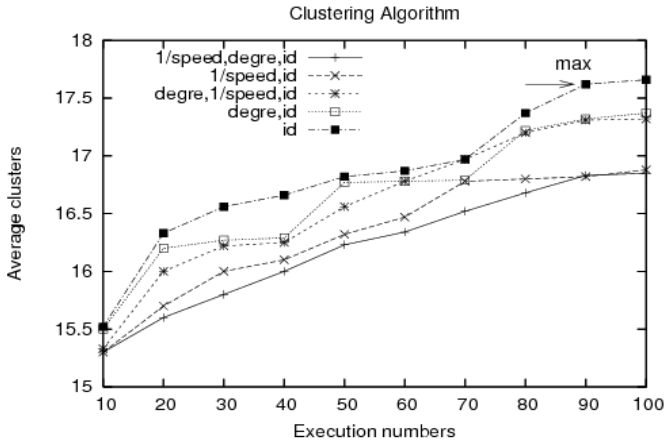


FIG. 3.8 – Mobilité basse

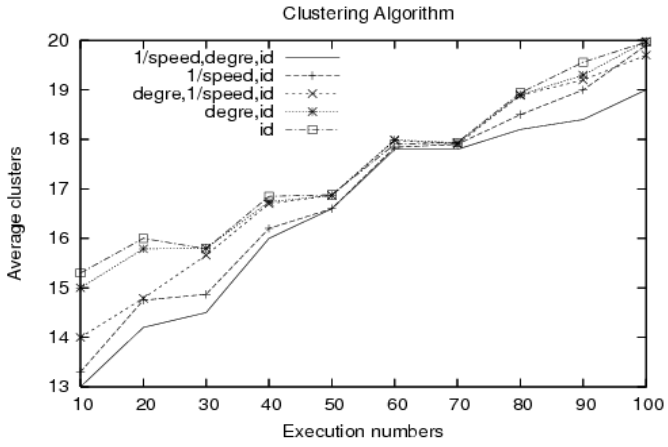


FIG. 3.9 – Mobilité haute

6 Conclusion

Dans ce chapitre, nous avons présenté différentes techniques de partitionnement. Le point de divergence principal entre elles est l'heuristique utilisée pour déterminer quel nœud sera le *clusterhead*. En effet, certaines se basent sur les identifiants des nœuds, d'autres sur leur degré, tandis que d'autres préfèrent utiliser un système de poids générique, dépendant d'un paramètre lié à la mobilité. Très clairement, ce système a l'avantage d'être plus proche de la réalité du réseau que les autres solutions. Le problème est désormais de formuler un bon système de poids permettant d'installer une structure de clusters la plus stable possible. Le système de poids choisi doit dans la mesure du possible refléter l'état du réseau.

L'état du réseau est sa simple description à des instants donnés par un certain nombre de paramètres qui peuvent être liés à la mobilité, au trafic, à l'énergie résiduelle, à la densité etc...Le problème qui se pose est d'obtenir un système de poids où l'unicité est garantie. Dans les propositions actuelles de partitionnement, on se contente de supposer cette unicité quelle que soit l'heuristique choisie, on s'aperçoit rapidement que ce n'est absolument pas garantie. Nous avons proposé un système de poids sans unicité de ce dernier afin d'obtenir une structure de clusters fiable. En effet, disposer d'une structure de clusters fiable permet d'améliorer très nettement l'implémentation de nombreux protocoles dédiés aux réseaux mobiles ad hoc (routage, initialisation, etc.). Toutefois, pour qu'on puisse mettre au point des algorithmes efficaces au-dessus des clusters, il reste fondamental d'améliorer notre technique de partitionnement qui présente encore quelques inconvénients, notamment l'obligation de faire un compromis entre fiabilité, réactivité et connectivité.

Chapitre 4

Geocasting sans GPS dans les réseaux de capteurs

1 Généralités

Un réseau de capteurs est avant tout un réseau de type ad hoc défini comme étant une collection dense d'unités (mobiles) déployées dans un environnement quelconque. Il est caractérisé par une topologie dynamique lorsque les unités sont mobiles, d'une bande passante limitée et des contraintes d'ordre énergétique. Le Geocating ou diffusion géographique consiste à envoyer des paquets de données d'un nœud source à un groupe de nœuds situés dans une région géographique spécifiée. L'idée est de s'appuyer sur les coordonnées géographiques des nœuds pour livrer le(s) paquet(s) à la bonne destination.

On peut au passage noté que plusieurs applications de ces types de réseaux sont basées sur la connaissance des positions des capteurs, positions qui peuvent d'ailleurs leurs être fournies par un dispositif de localisation comme le GPS. Mais cette solution reste à désirer lorsqu'on a affaire à une centaine voire des milliers de capteurs. Bien sûr que l'utilisation d'une solution telle que le GPS n'est pas envisageable dans toutes les situations. La puissance du signal est tellement faible qu'une partie en est absorbée par les couches atmosphériques de même que les feuilles des arbres. Nous imaginons facilement pourquoi une telle solution n'est pas exploitable à l'intérieur de bâtiment. De plus même en milieu ouvert où l'installation d'une solution satellitaire est possible, équiper des capteurs avec des récepteurs appropriés en augmenterait le prix, la consommation en énergie et le poids. Le contournement de telles solutions est alors souhaitable. Une autre approche devient alors nécessaire.

En s'appuyant sur les techniques existantes d'évaluation de la distance (`Time of Arrival` (ToA) [46], `Time Difference of Arrival` (TDoA) [7], `Receive Signal Strength Indicator` (RSSI) [45]) nous proposerons une méthode de localisation libre : `L-Libre`. Dans `L-Libre` les capteurs déterminent leur position en ne s'appuyant que sur les informations locales pour construire une système de coordonnées global. Ce système de coordonnées global va servir de support pour implémenter un nouveau protocole de geocasting qui garanti la livraison des paquets. Nous étudierons en première partie dans ce chapitre les techniques existantes de localisation dans les

réseaux ad hoc APS [3], GPS-less [40] et GPS-free [52]. En deuxième partie notre approche sur le principe de la libre localisation (sans GPS). La troisième partie va décrire notre nouveau protocole de geocasting avec garantie de livraison. Nous finirons enfin par les résultats de simulation pour justifier l'efficacité de notre démarche.

2 Les techniques de localisation

La localisation d'un nœud consiste en l'obtention ou le calcul de ses propres coordonnées via un périphérique dédié ou des méthodes de calcul distribué. Les coordonnées que les nœuds vont obtenir ou calculer peuvent être définies soit dans un système de coordonnées terrestre global, soit dans un système indépendant et virtuel. Les méthodes de positionnement actuels peuvent être regroupées en deux familles : la famille des méthodes basées sur les mesures physiques comme la distance, la puissance du signal reçu qu'on va appeler méthodes physiques approximatives et la famille des méthodes basées sur des mesures topologiques, comme par exemple le nombre de sauts entre deux nœuds appelées méthodes topologiques approximatives.

2.1 Les méthodes physiques approximatives

Parmi les solutions qui ont été proposées (Ad hoc Positioning System (APS) [3], méthodes combinant APS et angles d'arrivée [44], et d'autres [2, 7, 20, 24]) certaines utilisent, pour le calcul de la position la puissance du signal reçu (RSSI [45]) d'autres l'angle de réception (AoA) [44] et d'autre encore la différence de temps d'arrivée (TDoA) [7]. Nous décrivons brièvement quelques unes de ces techniques.

L'angle d'arrivée

Les méthodes basées sur l'angle d'arrivée Angle of Arrival (AoA) [44] du signal considèrent que chaque nœud est doté d'une antenne directionnelle lui permettant de découvrir l'angle qu'il forme avec le voisin émetteur. Elles considèrent également que certains nœuds appelés landsmarks (ou ancres) connaissent leur position.

Le principe de AoA est décrit comme suit : si nous connaissons les positions des trois sommets d'un triangle et les angles selon lesquels un point intérieur à ce même triangle voit les sommets, alors nous pouvons déterminer la position de ce point. Ce problème est plus souvent connu sous le nom de triangulation et est similaire à celui de la trilatération (basée sur les distances) utilisée dans le système de positionnement GPS. Si dans la figure 4.1, un nœud quelconque X, connaît en plus des positions des nœuds A, B et C, ses distances respectives a, b, c, à ces trois nœuds, alors il peut calculer ses propres coordonnées. De la même façon si ce nœud connaît les angles $\widehat{BXA}$, $\widehat{AXC}$, $\widehat{CXB}$ (respectivement α, β et γ) alors il est capable de connaître sa position en utilisant une triangulation. Cependant, comme les angles obtenus ne sont pas précis, AoA permet le calcul d'une zone de confiance plutôt que d'un point exact. De plus, le faible taux de nœuds connaissant leur position ne permet pas à l'ensemble de la topologie d'être dans le voisinage de trois nœuds

ancres. Ainsi, le calcul des positions s'effectue de saut en saut. Une fois qu'en nœud a calculé sa position, il peut servir comme nœud ancre pour que ses voisins calculent la leurs. On peut tout de même noté au passage que certaines méthodes basées sur la `TDoA` permettent de s'affranchir des antennes directionnelles, en proposant d'autres équipements alternatifs comme ceux développés au sein du `projet Criket du MIT` [19]. La méthode consiste à utiliser deux signaux. L'expéditeur envoie deux signaux RF(`Radio Frequency`) et ultrason en même temps et le calcul se fera sur la différence entre les temps d'arrivée de ces deux signaux. Deux récepteurs séparés d'une distance connue sont également nécessaires pour calculer l'angle d'arrivée du signal.

Temps d'arrivée

Le positionnement en utilisant le temps d'arrivée, `Time of Arrival` (ToA) [46] des messages est basé sur l'utilisation du temps écoulé entre le moment où une requête est envoyée vers un autre nœud et le moment où la réponse arrive. Le temps écoulé est composé de la somme du temps de propagation du signal dans les deux directions avec le temps de traitement au niveau du nœud récepteur. Si ce temps de traitement est connu à l'avance, alors le temps de transmission du signal est équivalent à la moitié du temps nécessaire à la transmission dans les deux sens. Si deux récepteurs supplémentaires sont disponibles, alors le calcul de la position est possible. Cette technique, en situation réelle, peut rencontrer quelques problèmes, comme l'estimation du délai de réponse. La raison principale en est les variations de conception des différents fabricants.

Une autre technique similaire et basée sur les temps d'arrivée est `Time Difference of Arrival` (TDoA) [7]. Cette technique estime la différence des temps d'arrivée des signaux d'une source vers plusieurs destinations. Ceci est obtenu par une vision globale du signal à un instant donné au niveau de plusieurs récepteurs. Une corrélation des versions reçues du signal au niveau des récepteurs est effectuée et son point culminant donne la différence des temps d'arrivées du signal. Une valeur particulière de la différence de temps d'arrivée définit une hyperbole entre les deux récepteurs entre lesquels l'émetteur peut se trouver. Si cette procédure est effectuée avec un troisième récepteur, alors une deuxième hyperbole est définie. Son intersection avec la première est une estimation de la position du nœud émetteur.

Cette technique comparée à `ToA`, possède l'avantage de ne nécessiter aucun périphérique spécifique supplémentaire, ni de synchronisation entre les nœuds.

Puissance du signal

Le système de localisation RADAR [45] mesure la puissance du signal et du rapport signal/bruit pour calculer les coordonnées des nœuds. Deux versions de RADAR existent : la première est basée sur une connaissance globale des signaux émis et une seconde sur une méthode de trilatération. Dans la première version, une cartographie des différents réseaux existants au niveau de la topologie est établie. Lorsqu'un nœud cherche à se localiser, il compare les caractéristiques des signaux qu'il reçoit avec celles de la cartographie et en déduit sa position.

La seconde version utilise les différences entre les puissances d'émission du signal

et sa puissance de réception pour déterminer la distance qui sépare la source de la destination. Avec un minimum de trois ancres dont le nœud connaît la position, il peut calculer, via une trilatération, sa position. RADAR arrive à obtenir des erreurs de placement respective de 3 et 4.3 mètres avec ces deux versions. Bien que les résultats de la première version soient meilleurs que ceux de la seconde, elle nécessite une reconstruction régulière de la cartographie. En effet les conditions météo, le déplacement de mobilier ou de groupes de personnes changent le principe de propagation des signaux, rendant ainsi la cartographie obsolète.

2.2 Les méthodes topologiques approximatives

Les méthodes de positionnement topologiques ne se basent que sur des caratéristiques de la topologie pour construire un système de coordonnées. Parmi les caractéristiques utilisées, nous pouvons citer le voisinage et la distance en nombre de sauts avec d'autres nœuds.

Nous décrivons ici deux méthodes de positionnement basées sur ce principe : DV-HOP et VCAP.

DV-HOP

Cette technique de positionnement DV-HOP (`Distance Vector-Hop`) [49] utilise des ancres (landmarks) et les distances entre les ancres sont connus d'avance. On peut décrire le principe en trois étapes :

1. Il utilise la technique du vecteur de distance et chaque nœud du réseau essaie d'obtenir sa distance exprimée en nombre de sauts par rapport à chaque ancre et ne conserve que le plus court chemin. Chaque nœud dispose d'une table $\{X_i, Y_i, h_i\}$. Chaque entrée de la table fait référence à une ancre et chaque nœud la met à jour en échangeant ses données avec seulement ses voisins directs.
2. Chaque ancre évalue la distance moyenne c_i qui correspond à la longueur en mètre d'un saut. Cette distance moyenne est enfin diffusée à tous les nœuds du réseau.

$$c_i = \frac{\sum \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}}{\sum h_i}, i \neq j \tag{4.1}$$

3. En recevant cette distance moyenne (c_i) chaque nœud peut, alors estimer sa position. La figure 4.2 en illustre un exemple.

VCAP

Les concepteurs de VCAP [1] ont proposés un algorithme de construction de système de coordonnées à trois dimensions sans aucun recours à une information de localisation. Cet algorithme est basé sur les distances en nombre de sauts entre les nœuds seulement. Il n'utilise aucune autre mesure. VCAP considère un réseau composé d'un grand nombre de capteurs, uniformément distribués dans la topologie, statiques ou très peu mobiles. Tous les nœuds sont supposés avoir le même rayon de couverture et des communications bidirectionnelles. Une fois le réseau déployé, trois

nœuds ancres sont choisis parmi l'ensemble de la topologie. Ces trois ancres sont choisis en fonction de leurs positions. En effet, ils doivent être placés en bordure du réseau, à égales distances les uns des autres. Ils ont proposé une méthode permettant de choisir des nœuds avec de telles caractéristiques. Ces trois nœuds inondent, chacun leur tour, le réseau avec des messages permettant à chaque nœud de découvrir sa distance en nombre de sauts, à ces trois ancres. Ces distances aux trois nœuds ancres vont être utilisées directement comme les coordonnées des nœuds dans un espace à trois dimensions.

La plupart des méthodes citées ci-dessus ne tiennent pas compte des caractéristiques bien spécifiques des capteurs, qui sont de petits périphériques communicants avec de faibles capacités, notamment énergétiques. Pour ces raisons et même si ces méthodes donnent des résultats satisfaisants et précis, il s'avère nécessaire de disposer de méthodes simples prenant en compte l'architecture spécifique des capteurs.

3 Localisation libre : `L-libre`

On part d'une charge utile dont dispose tout réseau de capteurs : le nœud puits (ou sink en anglais) afin de proposer une méthode de localisation sans GPS basée sur une stratégie l'utilisation de son antenne. Le sink constitue une passerelle entre le réseau de capteurs et le monde extérieur. Il collecte d'une part les données capturées par les capteurs afin de les acheminer vers un centre de traitement (ou utilisateur final) et d'autre part toute application utilisateur doit d'abord transiter par lui avant d'être soumise aux capteurs. On le qualifie de médiateur ou chef d'orchestre. Il dispose d'une antenne omnidirectionnelle avec des transmissions pouvant couvrir l'étendue du réseau, donc capable de diffuser une information à l'ensemble des capteurs. Par contre seuls les capteurs comptant le nœud sink dans leur voisinage peuvent communiquer directement avec celui-ci. Le réseau doit être multi-sauts afin de permettre aux autres capteurs ne comptant pas le nœud sink dans leur voisinage d'entrer en communication avec lui. Les capteurs doivent eux-mêmes estimer leur position par rapport au nœud sink comme le montre la figure 4.1.

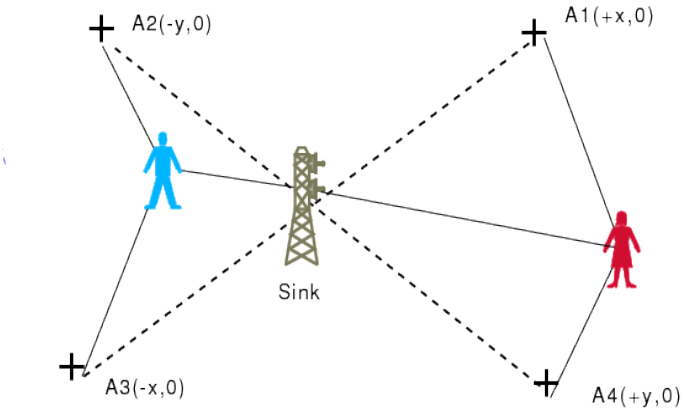


FIG. 4.1 – Estimation de position par rapport au sink

3.1 Définition du modèle

On dispose d'un réseau de n capteurs homogènes répartis dans une région géographique noté X, le rayon de transmission r est identique pour tous les capteurs. Du côté des capteurs se trouve un nœud particulier dénommé *sink* muni d'une antenne omnidirectionnelle dont le fonctionnement simule quatre antennes directionnelles (de 90^o) pouvant émettre indépendemment sous quatre angles comme le montre la figure 4.2. Le sink simule l'existence d'une ancre fictive A_i à chaque quart de tour de l'antenne omnidirectionnelle. Un système de coordonnées en 2D est établi au niveau du *sink* qui se situe au centre des quatre ancres A_1, A_2, A_3, A_4 placées respectivement sur les demi-droites (Ox^+)[1], (Oy^+), (Ox^-) (Oy^-) . On affecte au *sink* les coordonnées $(0,0)$, et les coordonnées des ancres A_1, A_2, A_3, A_4 sont respectivement $(x,0)$, $(0,y)$, $(-x,0)$ et $(0,-y)$ avec x, y connus du *sink* et $x \neq y$. On note D le diamètre du réseau déployé dans X et λ l'ensembles des ancres y compris le nœud *sink*. $N_u(\lambda)$ est l'ensemble des ancres immédiates dont les distances réelles sont connues par le nœud u. On exprime par d_{uv}, la distance entre deux capteurs quelconques u, v. Les coordonnées d'un capteur u sont notées (x_u, y_u).

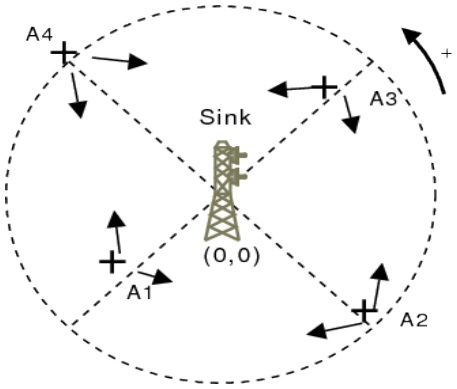


FIG. 4.2 – Référentiel du système de coordonnées

3.2 Technique d'estimation de distance

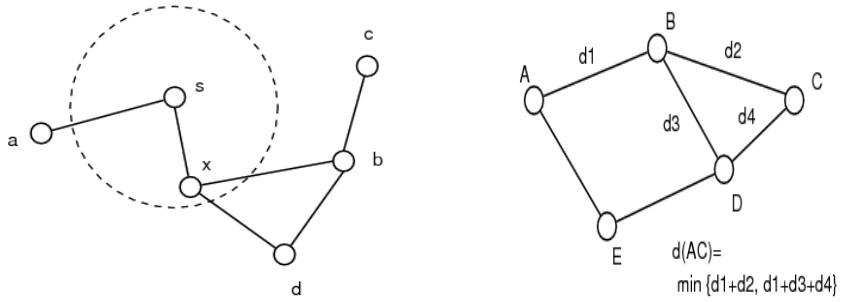
Pour évaluer la distance entre deux capteurs voisins en s'appuie sur l'une des techniques d'estimation de distance connue sous le nom de `Sum-dist` dont le principe peut se résumer ainsi : Chaque ancre émet un message contenant les paramètres suivants : son identité ,ses coordonnées et la longueur du chemin initialisée à zéro. Un nœud u qui reçoit ce message évalue la portée de l'émetteur, ajoute cette portée à la longueur du chemin et diffuse le message à ses voisins. Chaque nœud obtient ainsi sa distance par rapport aux ancres.

Il faut noter que seul le plus court chemin est retenu par chaque nœud. Dans la figure 4.3 , la distance estimée entre S et D est donnée par $d_{sx} + d_{xd}$ avec $d_{sd} \leq d_{sx} + d_{xd}$, relation liée à l'inégalité triangulaire. Soit $u_1, u_2, \ldots, u_q, a_i$ le chemin allant du nœud u à l'ancre A_i avec $A_i \in \lambda$,la distance estimée peut être définie récursivement de la manière suivante :

[1] désigne la demi droite des x positifs

$$\hat{d}_{u_1 a_i} = d_{u_1 u_2} + \hat{d}_{u_2 a_i} \tag{4.2}$$

avec $\hat{d}$ l'estimation obtenue avec $Sum - Dist$.



(a) (b)

FIG. 4.3 – Sum-Dist

Dans notre modèle lorsqu'un nœud u reçoit la position d'une ancre A_i, il estime la distance à cette ancre par la méthode de $sum - dist$. Cependant lorsque l'ancre A_i est voisine du nœud u, la distance réelle $d_{A_i u} \leq r$ est connue par u et u déduit qu'il se trouve sur le cercle de rayon $d_{A_i} u$ centré en A_i. Si A_i n'est pas voisine de u alors u en déduit qu'il ne se trouve pas dans un cercle de rayon r centré en A_i mais dans un cercle de rayon $\hat{d} A_i u$ (distance obtenue par sum-dist) centré en A_i. Cette distance estimée $\hat{d}_{A_i} u$ est telle que $d_{A_i u} \leq \hat{d}_{A_i u}$.

Cette technique est appliquée à chaque position d'ancre reçue et lorsque le nombre de trois ancres est atteint, le nœud u peut alors estimer sa position sur l'intersection de trois cercles centrés aux trois ancres. L'intersection deux à deux des trois cercles définit trois points distincts a, b, c. En utilisant le triangle a, b, c la position estimée de u sera son centre de gravité u'. Sur la figure 4.4 le nœud u cherche à estimer sa position en u', en recevant les positions des ancres A_1, A_2, $sink$, il estime les distances $\hat{d}_{A_1 u}$ (d^ (A1,u)), $\hat{d}_{A_2 u}$ (d^ (A2,u)), $\hat{d}_{sink,u}$ (d^ (sink,u)) avec la méthode $sum - dist$. Puisqu'aucune des ancres n'est voisine directe de u alors u conclut qu'il se trouve dans l'intersection des trois cercles de rayons $\hat{d}_{A_1 u}$, $\hat{d}_{A_2 u}$, $\hat{d}_{sink,u}$ centrés respectivement en A_1, A_2, $sink$. Cette intersection définit un triangle a, b, c et u estime sa position comme étant le centre de gravité de ce triangle.

Nous utilisons la propriété qui dit que le centre de gravité d'un triangle se trouve aux 2/3 de chaque médiane à partir du sommet. Si N est le milieu du segment [a,b], cette propriété se traduit par la relation vectorielle :

$$\overrightarrow{au'} = \frac{2}{3}\overrightarrow{an}$$

$$(x_n, y_n) = (\frac{x_a + x_b}{2}, \frac{y_a + y_b}{2})$$

$$(x_{u'}, y_{u'}) = \frac{2}{3}(x_n - x_a, y_n - y_a)$$

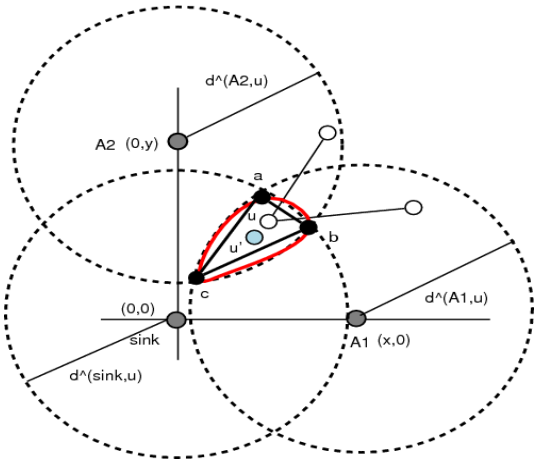


FIG. 4.4 – Estimation de la position de u à partir des ancres non voisines

Si ces trois ancres sont dans le voisinage immédiat du nœud u alors $\hat{d}_{A_1u} \leq r$, $\hat{d}_{A_2u} \leq r$, $\hat{d}_{sink,u} \leq r$ et ces distances estimées sont égales aux distances réelles d_{A_1u}, d_{A_2u}, $d_{sink,u}$. La position du nœud u sera l'intersection des trois cercles de rayons d_{A_1u}, d_{A_2u}, $d_{sink,u}$ centrés respectivement en A_1, A_2, *sink* et cette intersection se réduit en seul point $u' = u$ comme le montre la figure 4.5. Les coordonnées de u' sont obtenues en résolvant le système d'équations suivant :

$$\begin{cases} d^2_{sink,u} = (x_{u'} - x_{sink})^2 + (y_{u'} - y_{sink})^2 \\ d^2_{A_2,u} = (x_{u'} - x_{A_2})^2 + (y_{u'} - y_{A_2})^2 \\ d^2_{A_1,u} = (x_{u'} - x_{A_1})^2 + (y_{u'} - y_{A_1})^2 \end{cases}$$

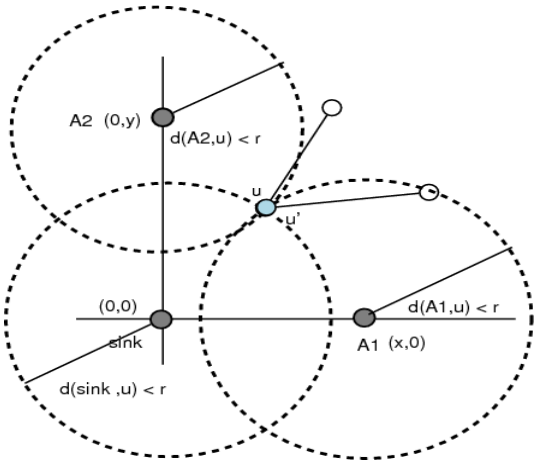


FIG. 4.5 – Estimation de la position de u à partir des ancres voisines

3.3 Gestion de l'antenne par le nœud sink

Le sink utilise son antenne omnidirectionnelle comme s'il possédait quatre antennes directionnelles (de 90^0 chacun). On considère quatre ancres fictives au niveau du sink dans le but de permettre à chaque nœud u du réseau de calculer sa position géographique. Chaque capteur u doit se servir de trois ancres y compris le

sink car il est considéré comme une ancre à part (c'est l'origine de notre référentiel terrestre) pour estimer sa position . Les distances $d_{sink,A_1}, d_{sink,A_2}, d_{sink,A_3}$ et d_{sink,A_4} sont connues. Le sink émet pour chaque ancre un message contenant les paramètres suivants : identité de l'ancre , coordonnées de l'ancre et longueur du chemin initialisée à zéro. Les informations d'un ancre A_i ne sont émises que dans le demi-plan qui le contient (par rapport au référentiel). Pour cela le sink va procéder en deux étapes :

1. découper le temps en quatre slots , et réserver à chaque slot une durée correspondant à la durée nécessaire pour acheminer un message entre les deux nœuds les plus éloignés (diamètre : D),

2. pour chaque slot S_i associer un angle α_i. La notation $\alpha_i(j,k)$ signifie α_i s'étend de j à k,

3. pour chaque angle α_i émettre ces informations (identité,coordonnées, longueur du chemin initialisé à zéro) et celles de l'ancre A_i.

Les angles directionnels sont calculés dans le sens trigonométrique de la manière suivantes : durant le slot S_i le sink émet les informations de l'ancre A_i sous l'angle $\alpha_i(i*\pi/2, i*\pi/2+\pi)$ ce qui nous donne la table de correspondance suivante :

Slot	Ancre	Direction de l'antenne
1	A_1	$\alpha_1(\pi/2, 3\pi/2)$
2	A_2	$\alpha_2(\pi, 2\pi)$
3	A_3	$\alpha_3(3\pi/2, 5\pi/2)$
4	A_4	$\alpha_4(2\pi, 3\pi)$

TAB. 4.1 – Les émissions directionnelles du nœud sink

3.4 Calcul des coordonnées du coté des capteurs

Dans cette section nous allons décrire les étapes nécessaire à la prise de décision par tout capteur afin d'estimer sa position. L'intersection de deux cercles définie deux points distincts M et M', l'idée est de permettre à tout capteur de déterminer s'il est en M ou en M'. Par exemple dans la figure 4.5 supposons que le capteur u cherche à déterminer sa position, les ancres A_1, A_2 sont des voisins immédiats de u par conséquent les distances d_{A_1u}, d_{A_2u} sont connues. Le capteur u déduit qu'il se trouve à l'intersection des deux cercles de rayons d_{A_1u}, d_{A_2u} centrés respectivement en A_1, A_2. Le capteur u peut alors se trouver en M $(i.e, (x_M, y_M))$ ou en M' $(i.e, (x_{M'}, y_{M'}))$. Rappelons que le nœud sink diffuse sa position et celle de chaque ancre A_i et lorsque un capteur quelconque reçoit la position d'un ancre A_i il estime sa position par la méthode $Sum-Dist$. Il reste cependant à enlever l'ambiguïté lorsque deux positions s'avèrent possibles pour un capteur comme le montre la figure 4.6. Nous définissons dans ce cas trois conditions pour la prise de décision.

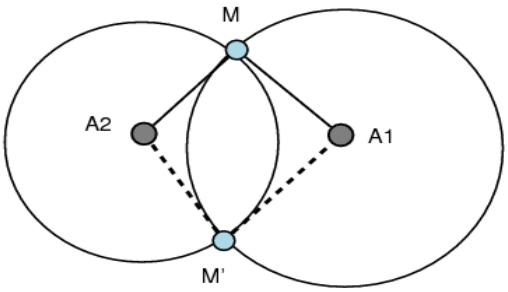


FIG. 4.6 – Ambiguïté de positions

Condition 1

La première condition consiste à définir une borne de la distance estimée (méthode $Sum-Dist$) de l'ancre par le capteur u qui cherche à se localiser lorsque l'ancre n'est pas un voisin immédiat. Sur la figure 4.7 l'ancre A_3 n'est pas un voisin direct de u ($A_3 \notin N_u(\lambda)$), donc u obtient la position de A_3 par une estimation.

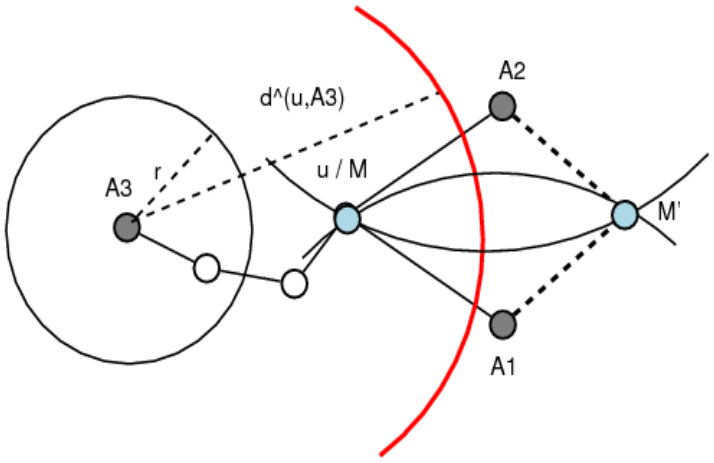


FIG. 4.7 – Condition 1

Le capteur u reçoit la position de A_3 (i.e (x_{A_3}, y_{A_3}) et estime sa position par rapport à A_3 (i.e $\hat{d}_{us_3}$). Le capteur u suppose être en M si les deux sous-conditions sont vérifiées :

1. $d_{MA_3} > r$ ce qui signifie que u et A_3 ne sont pas voisins.
2. $d_{MA_3} \leq \hat{d}_{uA_3}$ due à l'inégalité triangulaire

Si ces deux sous-conditions sont réunies alors u peut supposer être en M Cependant u est supposé être en M' si l'une des sous conditions n'est pas respectée. En conclusion u est en :

$$\begin{cases} M \ si \ r < d_{MA_3} \leq \hat{d}_{uA_3} \wedge (d_{M'A_3} \leq r \oplus d_{M'A_3} \geq \hat{d}_{uA_3}) \\ M' \ sinon \end{cases} \tag{4.3}$$

Condition 2

Ici aussi l'ancre A_3 n'est pas un voisin immédiat du capteur u qui cherche à se localiser illustré à la figure 4.8.

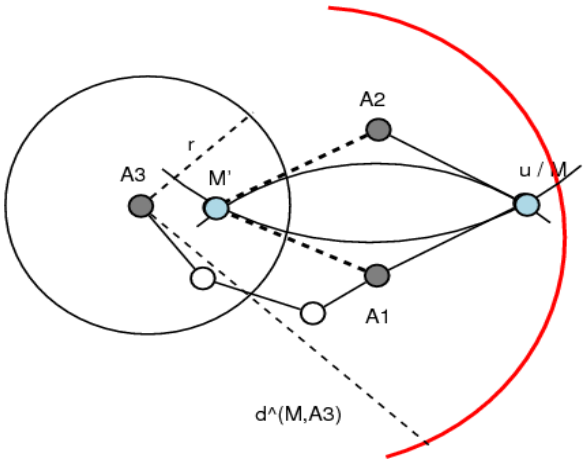


FIG. 4.8 – Condition 2

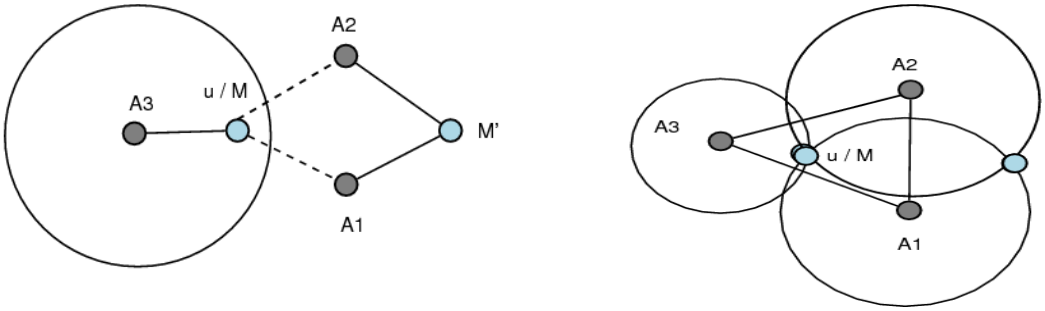
Quand u reçoit la position de l'ancre A_3 (x_{A_3}, y_{A_3}), il vérifie : si la distance réelle $d_{M'A_3} \leq r$ alors M' et A_3 seraient voisins. Dans ce cas u conclut qu'il n'est pas en M' mais en M :

$$\hat{d}_{MA_3} > r \wedge d_{M'A_3} \leq r \tag{4.4}$$

Cependant, si $\hat{d}_{MA_3} > r$ et $d_{M'A_3} > r$ alors u ne pourra pas conclure.

Condition 3

C'est le cas où le capteur u qui cherche à se localiser compte au moins trois ancres dans son voisinage immédiat. Chaque nœud est capable de mesurer les distances réelles par rapport à tout voisin immédiat.



(a) Trois ancres dans le voisinage immédiat

(b) trilatération

FIG. 4.9 – Condition 3

Ici A_1, A_2, A_3 sont des voisins directs de u, si u reçoit la position de l'ancre A_3, il vérifie : si $d_{M'A_3} > r$ alors M' et A_3 ne seraient pas voisins. Dans ce cas u conclut qu'il n'est pas en M' mais en M :

$$d_{MA_3} \leq r \wedge d_{M'A_3} > r \tag{4.5}$$

Cependant, si $d_{MA_3} \leq r$ et $d_{M'A_3} \leq r$ alors u ne pourra pas conclure.

Les distances obtenues n'étant pas des distances estimées le nœud u peut alors obtenir sa position par la méthode de trilatération. L'intersection des trois cercles centrés en A_1, A_2, A_3 sur la figure 4.9(b) donne la position du capteur u.

Un réseau de capteurs muni de moyen de localisation permet de connaître l'origine des données provenant des capteurs. On peut également se servir de cette notion de position pour désigner un certain nombre de capteurs pour une application spécifique. C'est cette possibilité de communiquer avec un groupe de capteurs désignés par leur position qu'on appelle le geocasting (diffusion géographique). La deuxième partie de ce chapitre est consacrée à l'étude de ce nouveau mode de transmission.

4 Geocasting

Le geocasting est un procédé qui consiste à envoyer d'un nœud source des données à un ensemble non vide de nœuds généralement des capteurs situés dans une région géographique désignée. Cette technique est souvent utilisée dans les réseaux de capteurs (dédiés à la surveillance) ; pour des cas suivants : lorsque le centre de surveillance a besoin de contacter tous les capteurs actifs au sein d'une *zone d'intérêt* pour recueillir des données périodiquement, ou de fournir un emplacement à des capteurs couvrant certaines zones pour des rapports d'évènements. Des méthodes d'inondations intelligentes existent pour cette tâche lorsque tous les capteurs actifs appartiennent à la *zone d'intérêt* mais lorsque la *zone d'intérêt* ne contient qu'un petit sous-ensemble de capteurs actifs l'inondation ne semble pas pratique, donc la tâche se réduit dans ce cas au *geocasting*. Les solutions existantes au problème du geocasting ne garantissent pas la livraison. Après un bref résumé des méthodes existantes, nous décrivons notre approche de geocating garantissant la livraison.

4.1 Définition du problème

Dans le problème traditionnel du multicasting les protocoles définissent un groupe multicast comme étant une collection d'unités inscrites à une même adresse appelée adresse du groupe multicast. Pour le geocasting le groupe consiste à un ensemble d'unités spécifiées dans une région donnée. L'acheminement des données d'une source S vers une région $\Re$ centré en D (D n'est qu'un nœud fictif pour matérialiser le centre de la région) peut se faire selon deux types d'algorithme de routage : un algorithme de routage par inondation modifiée et un algorithme de routage géographique. Ces algorithmes de routage résument le problème comme un simple routage entre une source S et une destination D comme le montre la figure 4.10 et les décisions de routage sont basées sur les positions des nœuds.

4.2 Algorithme de routage géographique

Un routage est dit géographique lorsque les décisions de routage sont basées sur la position des nœuds.

Les prérequis pour effectuer un routage géographique dans un réseau ad hoc sont :

- Les nœuds possèdent tous un moyen de localisation : soit un système natif comme le GPS, soit un système logiciel comme notre protocole L-Libre.

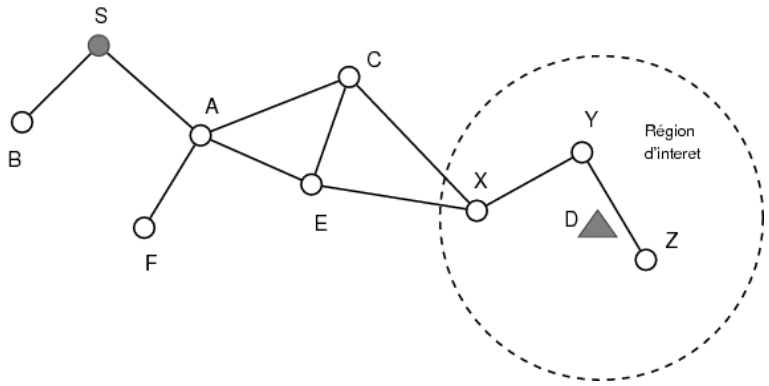


FIG. 4.10 – Exemple d'envoie geocast

– Un nœud source connaît toujours la position du nœud destinataire. Pour se faire, soit tous les nœuds connaissent les positions initiales de tous les nœuds, soit un service de localisation doit être utilisée.

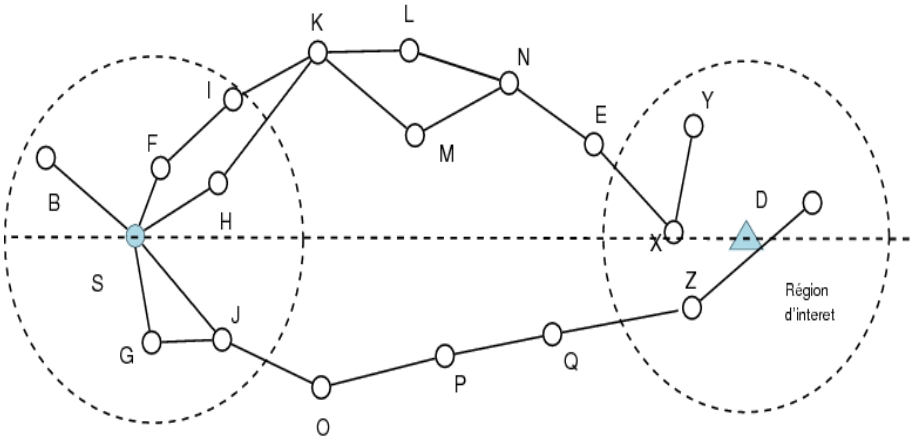


FIG. 4.11 – Différentes décisions de routage géographique

Considérons que le nœud source S désire router des informations vers le nœud destination D (figure 4.11). On distingue trois principales décisions de routage géographique :

Progression : La progression d'un nœud correspond à sa projection sur l'axe (SD). Le nœud dont la projection est la plus proche de la destination est choisie comme prochain nœud. Suivant le figure 4.10 le chemin résultant sera SIKLNEX. Cette approche fut la première approche de routage géographique, proposée en 1984 par Takagi et Kleinroch [30]. Elle est communément appelée `Most Forward with Radius` (MFR). Si la progression minimale est considérée, la méthode est appelée `Nearest Forward Progress` (NFP).

Distance : Le nœud le plus proche de la destination D en termes de distance est choisi comme prochain nœud. Le nœud J est sélectionné (figure 4.11) [18]. Si le nœud

possédant le message à transmettre est plus proche de la destination que tous ses voisins, il entre alors dans une phase critique d'échec de route. Une variante de cette approche,appelée Geographic Distance Routing (GEDIR) [35], consiste purement et simplement à éliminer un paquet s'il arrive dans ce cas critique. La route de S à D est SJOPQZ (figure 4.11).Dans ces approches, on cherche à minimiser le nombre de sauts. Une autre méthode consiste à prendre en considération le nœud le plus proche de la source en direction de la destination (`Nearest Closer` (NC)) [35] et le chemin est SGJOPQZ (figure 4.11). On utilise NC par exemple à des fins d'économie d'énergie.

Direction : Le nœud voisin le plus proche de la droite (SD) en direction de D est choisi [35]. Le nœud H est pris comme prochain nœud et le chemin entre S et D est SHKMNEX. On minimise dans ce cas la distance réelle parcourue par les messages entre les nœuds S et D. Une variante de cette approche est de considérer tous les nœuds en direction de la destination à un cône dirigé vers D.

Parmi les protocoles de routage géographiques les plus connus, on cite le *Geocast - Geographic Adressing Routing* (GGAR) [23], le *Greedy Perimeter Stateless for Wireless Networks* (GPSR) [29] et le *Geographical Routing Protocol* (GRP) [48]. Le *GGAR* est une réflexion basée sur une étude indiquant une possible utilisation des informations de géolocalisation en complément de l'adressage IP déjà existant [23]. Le routage dans un tel type de réseau est effectué de manière hiérarchique.

GPSR et *GRP*, proposent tous les deux une même approche globale de l'utilisation des informations de localisation. Si le nœud devant renvoyer l'information est le nœud le plus proche de la destination parmi tous les nœuds de son voisinage, nous sommes dans le cas d'un échec de route comme le montre la figure 4.12.

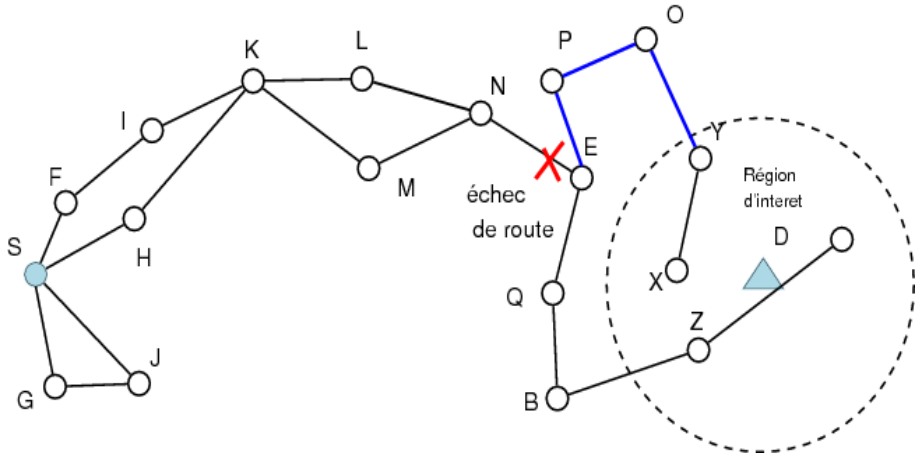


FIG. 4.12 – Échec de route pour le nœud E

GPSR et *GRP* qui avaient jusqu'alors le même algorithme de routage, divergent quant à leur approche pour résoudre ce cas particulier. GPSR préconise l'utilisation de l'algorithme de la *main droite*, algorithme arbitraire qui sélectionne uniquement les nœuds à droite du nœud source tandis que GRP ne fait aucune recommandation sur l'algorithme à employer. Le principal avantage de cette classe d'algorithme géo-

graphiques réside dans la capacité à trouver la meilleure route géographique possible pour chaque paquet émis, tout en ayant une vue restreinte du réseau ou n'ayant que des informations partielles de localisation. Enfin on peut citer des protocoles géographiques basés sur les *graphes de Gabriel* exposant une approche garantissant la livraison des paquets à la destination. Un graphe de Gabriel est un graphe planaire et sa construction rajoute des charges supplémentaires non négligeables.

4.3 Algorithme d'inondation modifiée

Il est probablement l'algorithme le plus simple pour délivrer des paquets multicast [31]. Il est implémenté comme suit : Si un nœud S désire envoyer un paquet multicast à l'ensemble des nœuds se trouvant dans la région spécifiée (geoscast region). Le nœud S est appelé émetteur et les nœuds F, G, Z et U constituent la région *geocast* ou *multicast*. Un nœud V recevant le paquet le transmet à ses voisins si c'est la première réception, dans le cas contraire il le jette. Sur la figure 4.13 lorsque le nœud X reçoit le paquet en provenance de B il le transmet à tous ses voisins mais s'il reçoit le même paquet de C, il le détruit.

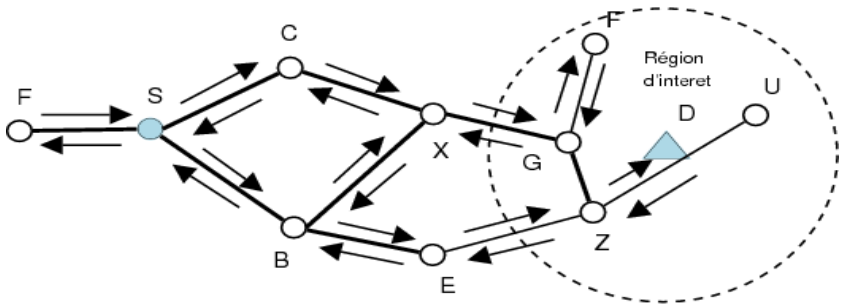


FIG. 4.13 – Inondation modifiée

Dans cette approche la garantie de livraison est assurée mais il se pose le problème suivant : Pourquoi autant de transmissions pour ne livrer des paquets qu'à un sous ensemble de k nœuds avec k largement inférieur à n lorsque celui-ci désigne la taille du réseau ? Une évaluation de la bande passante consommée en terme de sauts de cette approche peut nécessiter un coût s'élevant à $\sum_{i=1}^{n} d_i$ sauts avec d_i le degré du nœud i.

Des approches existantes au problème du geocasting avec garantie de livraison sont basées sur face routing [56] (routage basé sur les faces) et nécessite un graphe planaire. Impliquer les capteurs à la construction progressive de graphe planaire constitue une charge non négligeable pour ces équipements qui disposent d'un minimum de ressources.

4.4 Notre approche du geocasting

Nous venons de voir que la majorité des protocoles de routage basés sur des informations de position ne garantit pas la livraison à la destination. On part de la description de l'algorithme de routage DSR [31] pour y greffer des améliorations afin de réduire le nombre de messages de découverte de routes mais aussi de garantir

la livraison à la destination. Les informations de position qui sont nouveaux pour le protocole DSR permettront de délimiter un périmètre de recherche dans lequel l'algorithme de découverte de route sera plus efficace. Deux groupes sont alors à définir : La zone d'intérêt (région geocast) $\Re$ centré en D définit un cercle de rayon φ appelé *groupe multicast*. Un nœud u est membre du groupe multicast si $d_{uD} \leq \varphi$ avec d_{uD} la distance entre u et D. Le second groupe appelé *forwarding group* définit l'ensemble des nœuds relais (noté δ) autorisés à " *relayer* " les paquets multicast en direction de la région $\Re$, par définition $\Re \subset \delta$.

Les prérequis sont les suivants :

- Les nœuds possèdent tous un moyen de localisation par un système natif ou logiciel,
- Le nœud source connaît toujours la position du nœud destinataire,
- Le réseau est connexe.

Ainsi nous nous proposons simplement d'étendre le protocole de routage DSR afin de garantir la livraison à la destination. Il revient au nœud source qui désire envoyer un paquet multicast de définir le *forwarding group* δ en s'appuyant sur ces coordonnées et celles du nœud destinataire D. Le *forwarding group* ainsi défini recense tous les nœuds relais servant de support de communication entre S et les nœuds de la région d'intérêt $\Re$. la figure 4.14 donne une illustration des deux groupes.

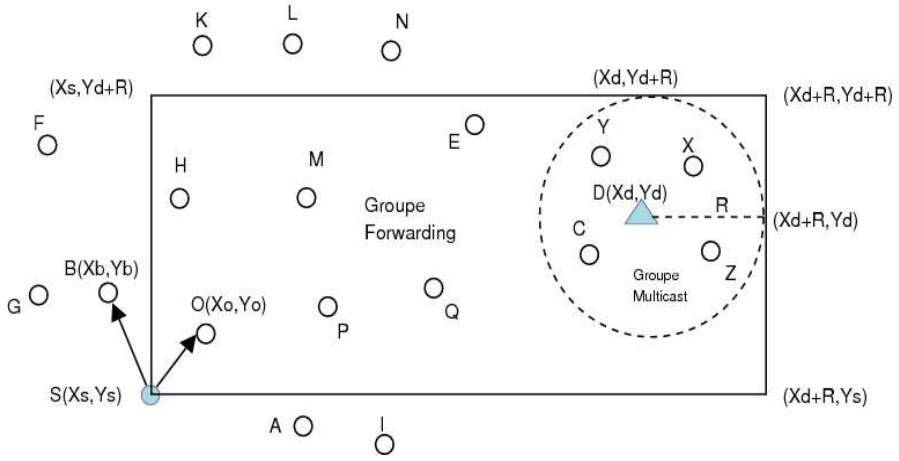


FIG. 4.14 – Groupes Multicast et Forwarding

Le protocole

Pour décrire notre protocole, nous allons d'abord donner une première proposition qui va permettre de recenser les cas pour lesquels la livraison n'est pas garantie et ensuite une seconde proposition va apporter les améliorations lorsque les nœuds sont en échec de route (cas de non livraison). Nous avons défini trois types de message de contrôle ou d'échange d'information . Ces messages sont sous la forme

de datagramme UDP : `Route_Request`, `Route_Reply`, `Position_Echange_msg`, `Impass_msg`. Les deux premiers ne sont rien d'autre que les types de messages de découverte de route du protocole DSR.

Propositions 1

Dans cette proposition, nous n'utilisons que les deux types de messages du protocole DSR (`Route_Request`, `Route_Reply`. Le message `Route_Request` est le message de découverte de route vers le groupe multicast $\Re$ de centre D. Le nœud S qui désire envoyer des paquets multicast en direction de cette région doit définir le *forwarding group* de façon explicite. Pour y parvenir la source S doit insérer dans le paquet `Route_Request` les coordonnées des quatre coins du rectangle. Lorsqu'une route est trouvée à l'issue de la procédure de découverte vers un au moins un nœud de la région d'intérêt, un paquet `Route_Reply` spécifiant cette route est envoyée au nœud source S. Le paquet `Route_Reply` est d'ailleur le premier paquet qui va emprunter cette route en sens inverse.

Pour trouver des routes les nœuds intermédiaires vont procéder de la manière suivante : lorsqu'un paquet `Route_Request` arrive à un nœud u, il vérifie d'abord si le paquet lui est destiné si c'est le cas(u appartient alors au groupe multicast) il répond à S par un paquet `Route_Reply` contenant le chemin à suivre. Si tel n'est pas le cas il teste s'il est autorisé à *relayé* (cas où $u \in \delta$). S'il est autorisé le message est *relayé* sinon il est détruit. La figure 4.15 donne un cas de succès de la proposition car les nœuds O, P, Q autorisés à *relayer* peut atteindre certains nœuds du `groupe multicast`. Les nœuds en dehors du *forwarding group* quant à eux détruisent systématiquement tout paquet `Route_Request`. Mais en cas d'échec de route ; situation où aucun nœud du groupe multicast n'est accessible à partir des nœuds autorisés à *relayer*, la proposition échoue systématiquement. Les figures 4.16 et 4.17 donnent les cas d'échec de cette première proposition.

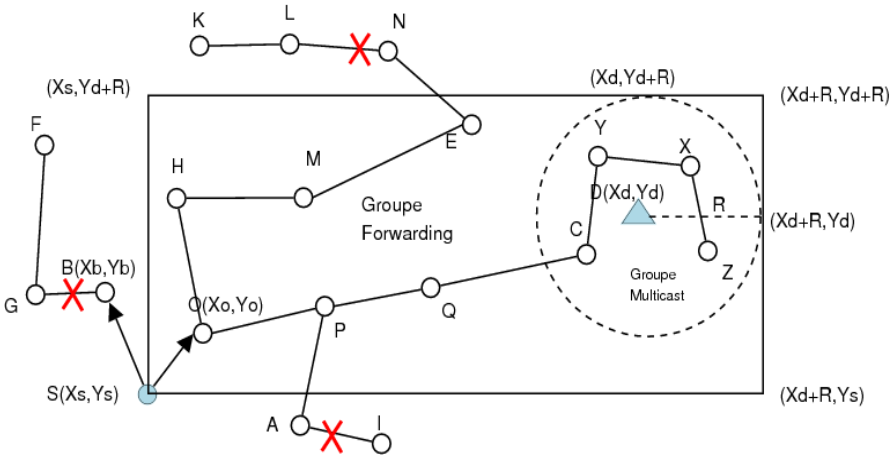


FIG. 4.15 – Livraison garantie vers la région Geocast

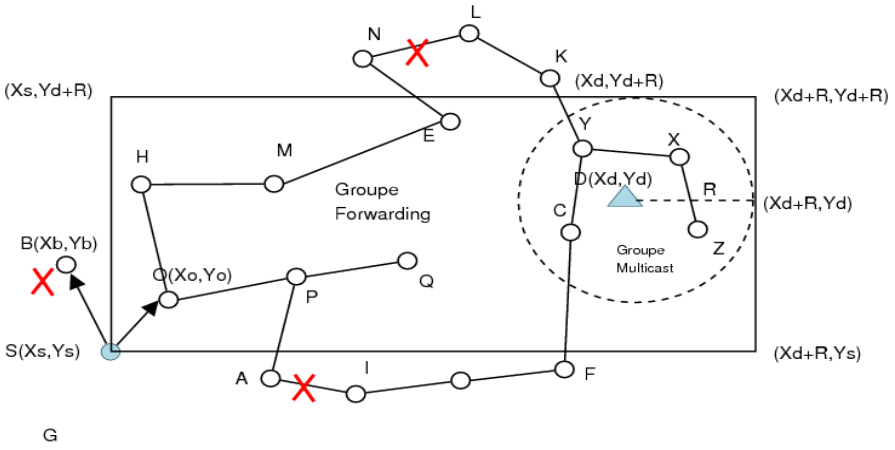


FIG. 4.16 – Situation d'échec : Nœuds intermédiaires en échec de routes

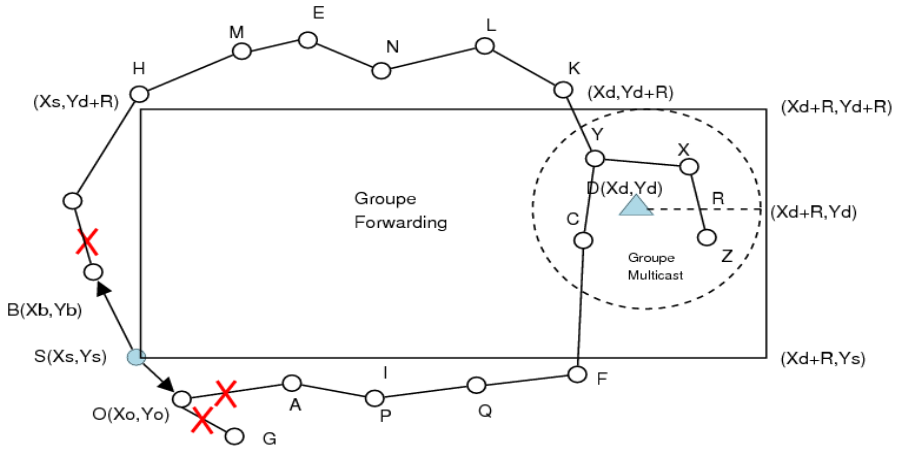


FIG. 4.17 – Situation d'échec : Aucun nœud intermédiaire

Propositions 2

Cette seconde proposition apporte à la précédente des fonctions supplémentaires afin de remédier aux cas d'échecs de routes. Nous avons alors introduit deux paramètres de dilatation d_1, d_2 et deux types de messages `Position_Echange_msg`, `Impass_msg`. Le réseau en question est matérialisé par une grille X de longueur L et de largeur l. Les différentes étapes mises en oeuvres dans cette proposition sont les suivantes :

1. **Étape 1** : le nœud source S qui désire envoyer des paquets multicast à une région $\Re$ de centre D s'appuie sur ses coordonnées et celles de D pour calculer le `forwarding group` δ de longueur L' et de largeur l' de sorte que $\Re \subset \delta$. Les parmètres de dilatation d_1 et d_2 sont initialisés à zéro lors de la première tentative de découverte de route par le nœud source S. Un paquet `Route_Request` définissant la source S, la destination D et le `forwarding group` est émis par la source pour trouver un itinéraire menant à D.

2. **Étape 2** : lorsqu'un nœud quelconque u reçoit un paquet de découverte de route `Route_Request`, il doit d'abord vérifier si le message lui est destiné : Si tel est le cas, il répond à S par un message de type `Route_Reply` en utilisant le chemin inverse du message `Route_Request` reçu. Sinon u doit recueillir les positions de tous ses voisins immédiats par un message de type `Position_Echange_msg`. Le message de type `Position_Echange_msg` ne contient que trois champs : identifiant de l'émetteur, coordonnées de l'émetteur et un flag (0 pour une demande et 1 pour une réponse). Ceci permet de recenser l'ensemble des nœuds voisins V' qui répondent au critère d'appartenir au `forwarding group`. Lorsque l'ensemble V' est non vide alors u *relaye* le message `Route_Request` à tout nœud $v \in V'$ comme le montre la figure 4.18. Le nœud u est en échec de route lorsque l'ensemble V' est vide. Pour remédier à cette situation u doit prévenir son prédécesseur (nœud à partir duquel le paquet `Route_Request` a été reçu) au moyen du message `Impass_msg`.

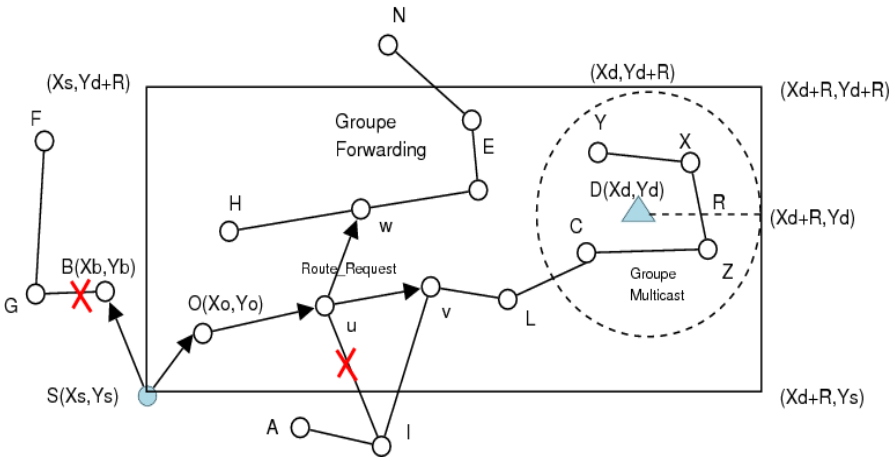


FIG. 4.18 – Le nœud u relaye aux voisins potentiels dans V'

3. **Étape 3** : lorsque v reçoit de u un message de type `Impass_msg` cela exprime le fait que u est en échec de route, v ne doit plus prétendre pouvoir passer par u pour atteindre la destination. Le nœud u est dans ce cas supprimer des voisins potentiels V' pouvant mener à D, en d'autre terme aucun message de type `Route_Reply` ne passera par u. Chaque réception par v d'un message de type `Impass_msg` fera l'objet de suppression du nœud émetteur w dans V'. Lorsque V' est vide, v sera à son tour en échec de route illustré à la figure 4.19, il doit aussi prévenir son prédécesseur au moyen du message de type `Impass_msg`. Cette manière de notifier un échec de route à ses prédécesseurs permet de remonter l'information jusqu'au nœud source S qui cherche une route vers D.

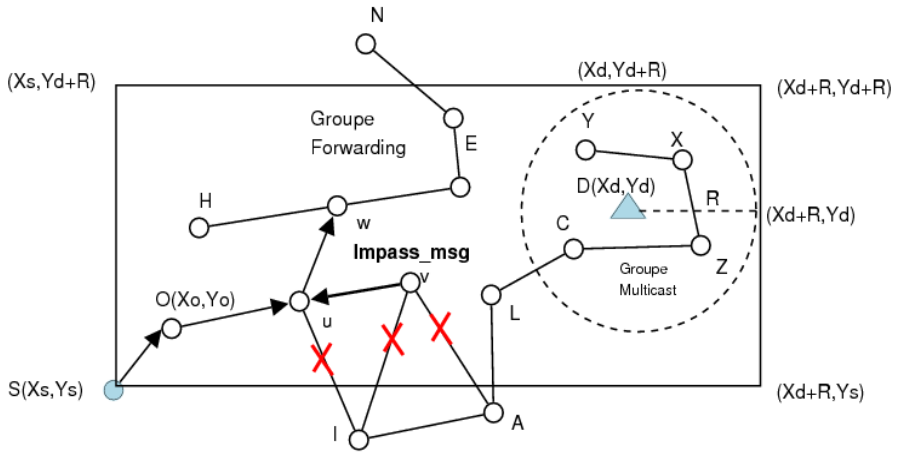


FIG. 4.19 – Notification par v d'un échec de route

4. **Étape 4** : lorsque S reçoit de v un message de type `Impass_msg`, il supprime systématiquement v des voisins potentiels V'. Si un message de type `Route_Reply` est reçu alors S retiendra ce chemin pour acheminer les données vers D et tous les autres messages de type `Impass_msg` seront ignorés. Lorsque tous les voisins immédiats de S répondent par un message de type `Impass_msg`, ils seront alors tous supprimés de V' et le nœud S sera en échec de route puisque V' est vide. Il faut alors redonner à S la possibilité de répartir à la découverte d'un nouveau chemin avec une légère modification de l'état précédent qui a provoqué un échec de route.

   Le `forwarding group` précédent doit être amélioré à l'aide de deux paramètres d_1 et d_2 nommés paramètres de dilatation comme le montre la figure 4.20. Soient n le nombre d'itérations souhaitées, d_1 est la marge à rajouter à la longueur L' du `forwarding group` et d_2 la marge à rajouter à la largeur l' du `forwarding group`. Les valeurs de ces marges sont $d_1 = \frac{L-L'}{n}$ et $d_2 = \frac{l-l'}{n}$. Tant qu'il n'existe pas de route vers la destination D, la source S doit réadapter la taille du `forwarding group`. Sur l'exemple de la figure 4.20 au bout de trois tentatives la source S a réussi à trouver un chemin qui mène à D. Dans le pire des cas notre proposition convergera vers l'algorithme d'inondation cité ci-dessus.

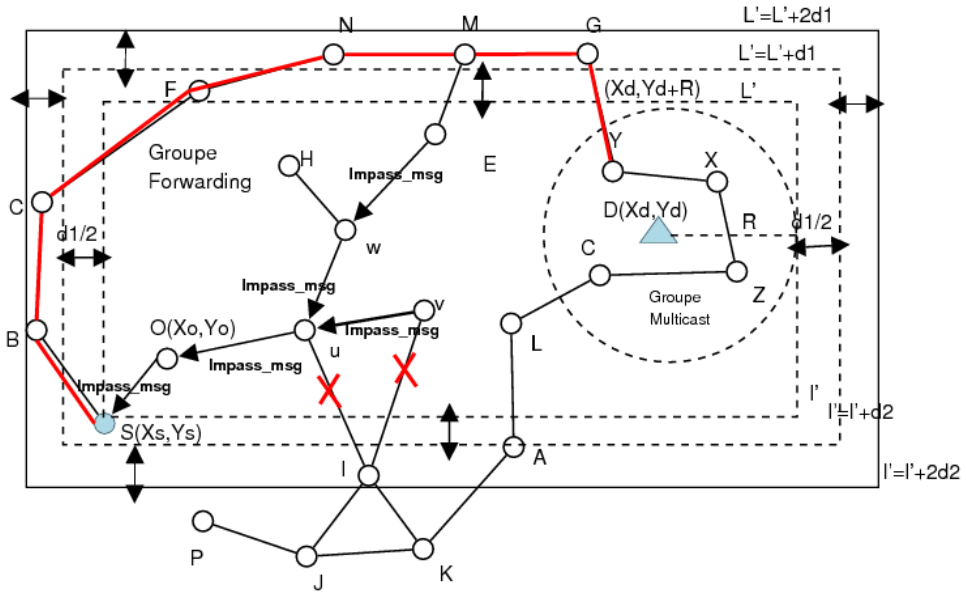


FIG. 4.20 – Notification d'échec de route par tout $v \in \delta$

4.5 Quelques résultats

Nous avons testé nos deux propositions de geocasting sur dix topologies de taille variable dans une grille de taille 100x100. La source S des paquets multicast reste inchangé et les capteurs sont homogènes. Aucune topologie ne dispose de capteur isolé (topologie connexe). Pour la première proposition, il existe toujours un chemin entre la source S et la destination (groupe multicast). La seconde proposition met en œuvre une adaptation de forwarding group en cas d'échec de route.

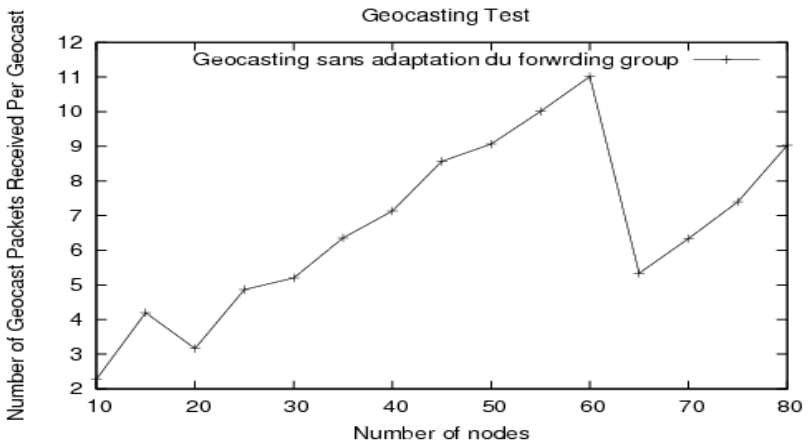


FIG. 4.21 – Nombre de paquets vs nombre de nœuds : Proposition 1

La figure 4.21 donne le nombre moyen de paquets multicast par capteurs sans adaptation du forwarding group. La source S et la destination restent inchangées. Le nombre de paquet multicast ne dépend pas de la taille globale du réseau mais de celle du forwarding group.

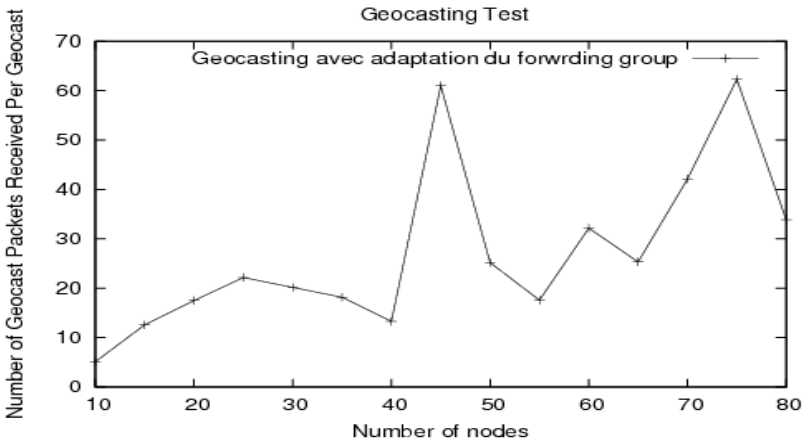


FIG. 4.22 – Nombre de paquets vs nombre de nœuds : proposition 2

Sur la figure 4.22 chaque pic correspond à une adaptation du forwarding group après un échec de route. Le nombre de paquets multicast peut soit augmenter ou même diminuer en fonction du nombre d'éléments présents dans le forwarding group. La figure 4.23 exprime l'écart en terme de paquets multicast entre nos deux propositions. La première proposition est préférable lorsqu'on dispose d'une connaissance complète du réseau tandis que la seconde est capable de se réadapter en fonction de la topologie et la livraison à la direction est garantie.

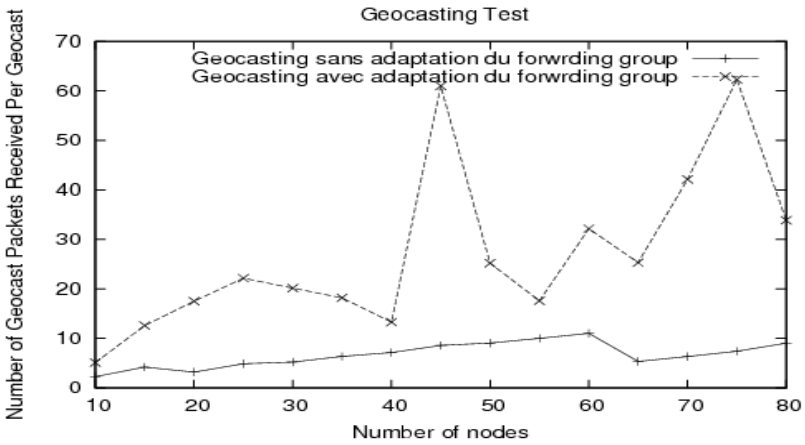


FIG. 4.23 – Proposition 1 vs Proposition 2

Dans tous les cas notre approche engendre moins de paquets multicast que l'inondation modifiée qui peut impliquer tous les capteurs lors d'une transmission (figure 4.23). Notre approche permet de réduire à chaque étape le nombre de capteurs relais qui doivent être impliqués, mais aussi d'adopter le comportement de l'algorithme d'inondation dans le pire des cas.

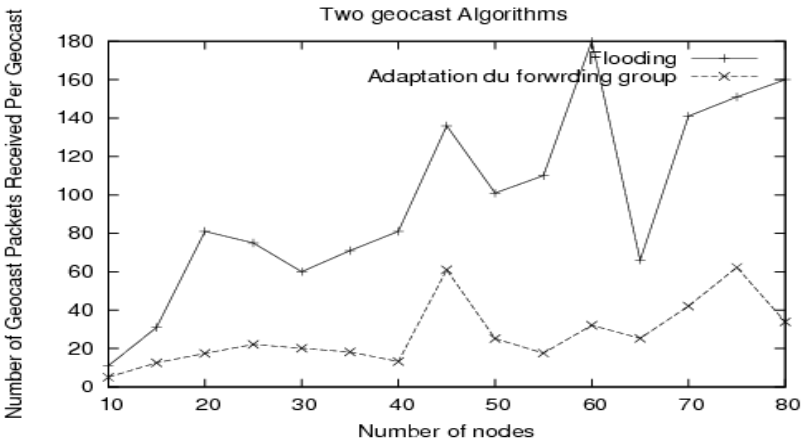


FIG. 4.24 – Proposition2 vs Inondation

L'approche adaptative que nous avons proposé permet de réduire le trafic lié à la découverte de routes. La figure 4.24 donne un aperçu de ce trafic comparé à l'inondation.

5 Conclusion

Nous avons proposé dans ce chapitre un algorithme simple d'estimation de position pour l'ensemble des nœuds du réseau avec seulement quatre ancres fictives, formant un système de coordonnées 2D centré au nœud *sink*. Notre algorithme `L-Libre` comporte trois étapes : la première consiste à faire parvenir à l'ensemble des nœuds les positions et les distances des ancres y compris le nœud sink. La seconde étape consiste pour chaque nœud de déterminer si les ancres sont des voisins immédiats auquel cas les distances réelles sont utilisées sinon les distances estimées sont utilisées. Chaque nœud doit également lever l'ambiguïté lorsque deux positions sont possibles. Enfin la dernière étape consiste à estimer la position. Nous avons proposé pour y parvenir une méthode mathématique de résolution d'équations (qui décrivent les situations).

Dans la seconde partie de ce chapitre nous avons proposé une approche du geocasting avec garantie de livraison à la destination. Notre algorithme s'inspire du traditionnel algorithme *DSR*. Nous avons rajouté à ce protocole deux autres messages (`Position_Echange_msg, Impass_msg` pour aider à la prise de décision lors d'un `échec de route` dans un domaine restreint (forwarding group δ) calculé à l'avance par le nœud émetteur. Le message de type `Position_Echange_msg` permet de collecter les informations de position du voisinage et le message de type `Impass_msg` permet de signaler un `échec de route` à un nœud prédécesseur. Lorsqu'un échec de route est signalé par tous les voisins directs du nœud émetteur S celui-ci doit réadapter le `forwarding group` pour procéder à une nouvelle découverte de routes.

Chapitre 5

Maintenance de la connectivité et tracking dans les réseaux de capteurs

1 Généralités

L'accès aux données à distance a toujours été une application dans les réseaux mobiles et fixes. Considérons un système de communication dans un champ de bataille constitué de soldats et de véhicules mobiles. Les unités avancent ensemble pour accomplir une tâche en formant un réseau ad hoc. Dans cet environnement les soldats devraient être en mesure de disposer des informations plus récentes mais aussi de les partager avec d'autres dans l'opération. Le problème classique causé par les nœuds en mouvement dans un réseau ad hoc est le partitionnement (décomposition en des sous ensemble connexes). Prévoir ces partitions peut être utile pour les applications dans un environnement ad hoc mobile. En effet, étant conscient d'un avenir de déconnexion du réseau peut aider à assurer une meilleure qualité de service par l'adaptation du comportemet des applications.

Les algorithmes actuels proposés pour résoudre ce problème s'appuyant sur des informations de position qui doivent leur être fournies par un système de positionnement. Ces informations de position doivent être mises à jour après chaque déplacement d'un ou de plusieurs nœuds et ceci ne fera qu'ajouter des charges supplémentaires au problème en question. Il semble indispensable de proposer d'autres approches qui ne tiendront pas compte de la position des nœuds dans le réseau. Dans la première partie de ce chapitre nous allons d'abord décrire les travaux existants sur le maintien de la connectivité et ensuite nous proposerons une méthode originale permettant de détecter à l'avance les partitions sans l'aide d'aucun système de positionnement afin de rendre accessible les données avant que ces partitions ne se forment. Nous exposerons une approche pour la poursuite de cible ou tracking dans les réseaux de capteurs en deuxième partie de ce chapitre.

2 Détection des partitions dans la littérature

Le réseau ad hoc par définition engendre des propriétés de connexité qui demeurent le plus souvent critiques. Le partitionnement du réseau ad hoc en plusieurs composantes connexes est lié à la mobilité non contrôlée des nœuds. Mener à terme de façon collaborative une application dans ces conditions semble impossible puisque

les informations relatives à l'application seront inaccessibles par tous. L'alternative proposée pour résoudre ce problème consiste à détecter les partitions en temps opportun (à l'avance) afin d'anticiper certaines tâches telles que, l'accès à tous aux données importantes. Prenons l'exemple de la figure 5.1 à un instant t 6 nœuds sont reliés les uns aux autres en formant un seul groupe g, le mouvement des nœuds et la direction de ces mouvements font qu'à l'instant $t + h$ le groupe se divise en deux groupes g_1 et g_2. Ceci fait que toute donnée importante tenue par un nœud du groupe g_1 (respectivement g_2) n'est plus accessible aux nœuds du groupe g_2 (respectivement g_1).

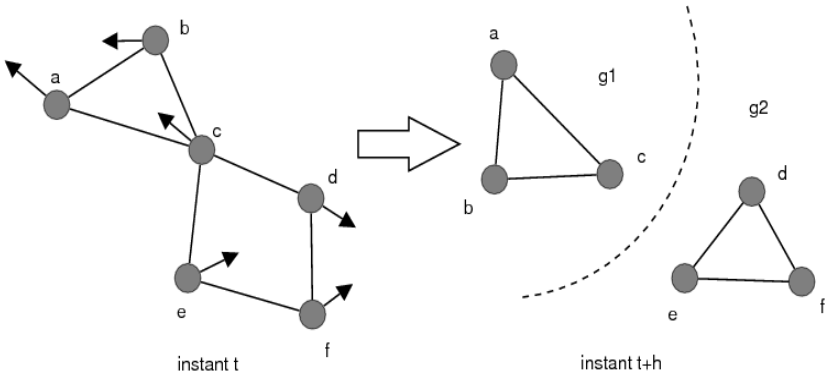


FIG. 5.1 – Décomposition en deux groupes

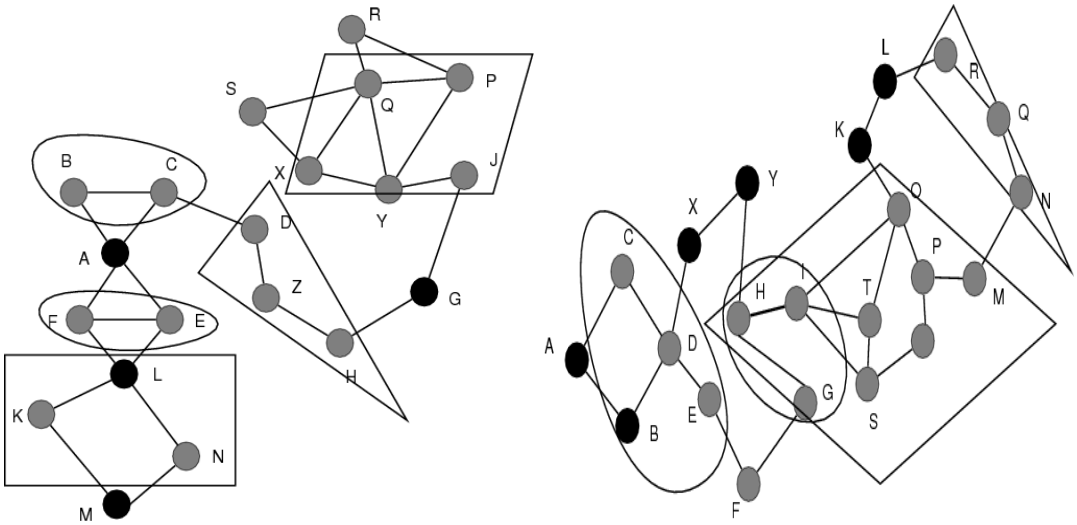
Dans [26] les auteurs ont proposé un service de reproduction des données qui consiste à copier les données du groupe g_1 sur certains nœuds du groupes g_2 et les données du groupe g_2 sur certains nœuds du groupe g_1 avant que le partitionnement n'ait lieu. Les auteurs s'appuient sur les positions courantes des nœuds (à l'instant t), leur vitesse de déplacement, leur direction et leur rayon de transmission pour prédire les futures positions. Une réévaluation des distances entre les positions actuelles et les futures positions des nœuds déterminera les partitions possibles de la nouvelle topologie. Le nœud qui a détecté la ou les partition(s) doit veiller à ce que les données de sa partition soient reproduites (ou copier) sur toutes les autres partitions. La valeur de h est choisie de telle sorte que l'opération de reproduction puisse être achevée avant l'instant $t + h$.

Wang et Li [33,34] ont proposé la même approche de façon centralisée en utilisant un algorithme de clusterisation s'exécutant sur un nœud central (serveur). Chaque nœud du réseau envoie au nœud central sa position et sa vitesse de déplacement. Connaissant la vitesse de déplacement et la position de chaque nœud, le serveur peut prédire les éventuelles partitions qui pourront entraîner des déconnexions et prévenir à temps les nœuds impliqués

Dans [38] les auteurs ont proposé une approche différente de celles énumérées ci-dessus. Leur idée était de repérer les nœuds et les liens (ou liaisons) qui peuvent provoquer des partitions. Ces nœuds et ces liens sont appelés respectivement nœuds critiques et liens critiques. Un nœud est dit critique lorsque son déplacement ou sa suppression entraînera l'apparition d'un ou plusieurs sous ensembles connexes. Un lien est dit critique lorsque les deux nœuds qu'il relie sont des nœuds critiques. Chaque nœud n'a plus besoin de connaître la topologie complète du réseau comme

les méthodes précédentes mais seulement un ensemble de voisins à une distance au plus $k-sauts$. La collecte des voisins à $k-sauts$ d'un nœud u définit un sous graphe g'_u.

Le nœud u est dit *k-critique* si et seulement si le graphe résultant de la suppression de u et des arêtes issues de u dans g'_u est non connexe, ceci n'est que la première version de leur algorithme. Ils l'ont nommé *topological critical node algorithm* puisque seules les informations topologiques permettaient à un nœud d'évaluer son statut (critique ou non). La deuxième version intitulée *positional critical node algorithm* n'est qu'une amélioration de la première dans laquelle les positions des nœuds sont connues en utilisant des GPS ou un autre système de localisation semblable. Un lien $(u \longleftrightarrow v)$ est critique : si les ensembles des voisins à $k-sauts$ de u et de v sont disjoints et qu'il n'existe pas un voisin commun aux deux ensembles. La figure 5.2 donne un exemple de détection des nœuds et liens critiques.



(a) A et M sont 1-critique, G est 3-critique

(b) HI,AB,XY,KL sont connus liens critiques

FIG. 5.2 – Exemples de nœuds et liens critiques

L'algorithme DFS (depth first search) a été utilisé dans [6, 16, 54] pour détecter les liens critiques. Il s'agit d'un algorithme centralisé qui exige à chaque nœud une connaissance globale ou complète de la topologie. Les messages de mise à jour des liens lorsque les nœuds sont en mouvement sont très coûteux et le rendent inefficace. Afin de retarder ou éviter la rupture d'un lien signalé critique, deux approches ont été proposées : la première consiste à changer toute trajectoire passant par ce lien en choisissant un nœud voisin des nœuds formant le lien critique. En seconde approche augmenter le trafic dans le réseau pour allonger la connectivité en supposant qu'aucun mouvement n'est autorisé lorsque le trafic est dense.

3 Notre approche de détection des partitions

Les méthodes énumérées ci-dessus ont prouvées leurs efficacités sur la détection des partitions critiques, sur les mesures à prendre en cas d'occurrence de partitions. Les approches centralisées sont lourdes en termes de messages sur des réseaux de type ad hoc de taille (nombres de nœuds) importante. Le nœud central doit périodiquement collecter des informations sur l'étendue de la topologie pour mettre à jour les liens de communication avant de prévenir les nœuds impliqués sur des éventuelles partitions. Les approches basées sur les positions géographiques des nœuds exigent l'utilisation de GPS ou d'un système similaire, pratiques mais ces informations de positions doivent être mises à jour après chaque déplacement d'un ou de plusieurs nœuds et ceci ne fera que ajouter des charges supplémentaires au problème en question.

Nous proposons d'abord une approche distribuée permettant de détecter à l'avance les partitions sans l'aide de tout système de positionnement. Et ensuite une méthode simple qui vise à copier les données entre partitions avant que ces dernières n'apparaissent physiquement.

3.1 Modélisation et description du problème

Soit le graphe non orienté $G = (N, E)$ représentant le réseau ad hoc, avec N l'ensemble des nœuds et E l'ensemble des liens de communications. Un nœud u est dit voisin d'un nœud v ($u \in \Gamma(u)$) lorsque u se trouve dans la zone de couverture de v et réciproquement. Pour une analyse simpliste on suppose que le rayon de couverture ou rayon de transmission r est identique pour tous les nœuds. Le graphe G peut être connexe, biconnexe, 3-connexe ou même k-connexe. Un nœud u est dit critique lorsque son déplacement ou sa suppression provoquera l'apparition d'un ou plusieurs sous ensembles connexes.

Dans la théorie des graphes cette définition correspond à la définition *d'un point d'articulation* ou point de jonction facilement repérable par un algorithme de parcours en profondeur d'un graphe (DFS). Nous mettons en oeuvre une version modifiée de l'algorithme DFS afin de détecter à l'avance les éventuels liens (liens critiques) qui peuvent scinder le réseau en un ou plusieurs composantes connexes. L'algorithme modifié DFS est exécuté en local sur une topologie partielle représentée par un sous-graphe G' qui identifie l'ensemble des voisins à *k-sauts* pour chaque nœud . Pour construire ce sous graphe G'_u chaque nœud u doit collecter ou identifier ses voisins à *k-sauts* au moyen de messages de type *hello*.

Notre proposition DFS permet à chaque nœud u de déterminer autour de lui tous les liens critiques et l'ensemble des partitions qui peuvent être causées par ces liens. On évalue également dans cet algorithme la taille de chaque partition. Lorsque le sous-graphe G'_u ne comporte aucun lien critique, on dit qu'il est biconnexe c'est à dire quelque soient le triplet u,v et w, il existe un chemin p reliant u et v et qui ne passe pas par w.

3.2 Algorithme DFS

DFS est un algorithme connu de parcours de graphe. Son exécution peut être représentée par un arbre enraciné au nœud à partir duquel il a été lancé. Soit le réseau connexe de la figure 5.3, en exécutant DFS à partir du nœud m, on obtient l'arbre de la figure 5.4 enraciné en m. Dans cet exemple les arêtes sont visitées selon l'ordre donné dans la table d'adjacence Tab 5.1. Les nombres sur la figure 5.4 donne l'ordre de parcours des nœuds par l'algorithme DFS. Les arcs en pointillés sont appelés arcs de retour, exprimant le fait qu'un nœud déjà visité a été rencontré de nouveau et les lignes en flèches sont les arêtes de l'arbre résultant du parcours DFS. La racine de cet arbre est dite point d'articulation si et seulement il possède plus d'un fils. Un sommet quelconque différent de la racine est dit pointe d'articulation lorsqu'il existe au moins un descendant non connecté à un ancêtre par un arc de retour.

Systématiquement tout point d'articulation peut être pris comme point ou nœud critique d'après la définition qu'on s'est donnée. Tout lien (arête) issu d'un nœud critique est logiquement un lien critique. Notre version DFS visite les nœuds (par marquage) de la même manière que le DFS classique. Le nœud qui exécute DFS est appelé nœud *racine*. Soit le nœud u, v et w des voisins immédiats de u, si u visite en premier le nœud v alors le lien $(u \longleftrightarrow v)$ est appelé *lien initial*. Si les voisins de v autre que u arrivent à atteindre (dans l'exécution de DFS) la racine u alors le lien $(u \longleftrightarrow v)$ sera considéré comme lien non critique puisque le nœud u peut être atteint par un lien autre que le lien $(v \longleftrightarrow u)$. L'algorithme nous retournera dans ce cas un nombre négatif. La version complète de notre algorithme est donnée ci-dessous. Le paramètre *size* permet de calculer la taille (en nombre de nœuds) des partitions éventuelles.

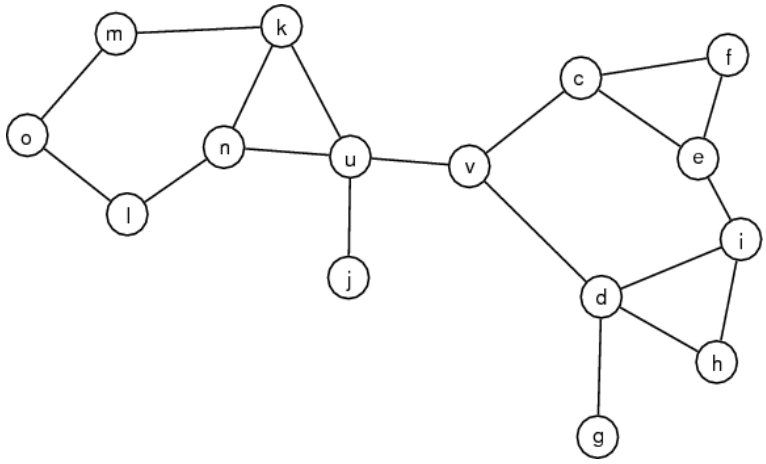


FIG. 5.3 – Graphe connexe ou mono-connexe

Nœud	Voisinage
c	v, e, f
d	g, h, i
e	c, f, i
f	c, e
g	d
h	d, i
i	d, e, h
j	u
k	u, m, n
l	n, o
m	k, o
n	u, k, l
o	l, m
u	v, j, k, n
v	u, c, d

TAB. 5.1 – Table d'adjacence du graphe 3.3

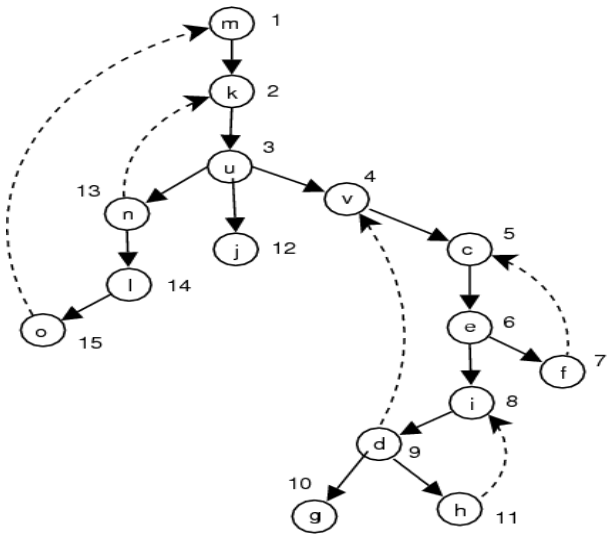


FIG. 5.4 – L'arbre correspondant au parcours DFS

Algorithm 7 DFS (v) : retourne un entier

Le nœud v cherche à vérifier si lien $(v \longleftrightarrow u)$ est critique ou non
Il marque u comme visité est lance DFS(u). v est le nœud la racine et $(v \longleftrightarrow u)$ le lien initial.

```
1: begin
2: Variables i, s, size : entier
3: s := 1
4: size := 1
5: for (chaque voisin u ∈ Γ(v) non marqué) do
6:     if (v,u) forme l´ arete initiale then
7:         continuer
8:     end if
9:     if u = racine then
10:        retourner −100
11:    end if
12:    if (u est nom marqué) then
13:        marquer u
14:        i = DFS(u)
15:        if (i < 0) then
16:            s = −1
17:        end if
18:        if (i ≠ −100) then
19:            size = size + abs(i)
20:        end if
21:    end if
22: end for
23: retourner s ∗ size
24: end
```

Cet algorithme détecte les éventuelles partitons et leur taille avec une complexité en $O(N + E)$ où N représente l'ensemble des nœuds et E l'ensemble des arêtes du graphe.

3.3 Détection des liens critiques

Comme nous l'avons décrit au paragraphe précédent l'exécution de l'algorithme DFS peut être matérialisée par un arbre enraciné au nœud appelant.

Chaque nœud u doit déterminer autour de lui tous les liens critiques en commençant d'abord par collecter ses voisins à *k-saut* (construction du sous-graphe G'_u). Ensuite exécuter l'algorithme DFS pour chaque voisin immédiat de la manière suivante :

1. Pour tout voisin v marquer le voisin comme *vu* ou *visité*
2. Lancer *DFS(v)* pour statuer le lien initial $(u \longleftrightarrow v)$

S'il existe un sommet interne pouvant atteindre par un arc de retour la racine de l'arbre qui décrit le parcours alors on en déduit que le lien $(u \longleftrightarrow v)$ n'est

pas critique puisqu'il existe un autre chemin qui mène à u sans passer par le lien $(v \longleftrightarrow u)$. Sinon on conclut que le lien initial $(u \longleftrightarrow v)$ est critique puisqu'il est le seul lien qui permet à u d'atteindre son voisin v . Sur la figure 5.6 aucun arc de retour ne permet d'atteindre la racine u en conclusion le lien $(u \longleftrightarrow v)$ est un lien critique.

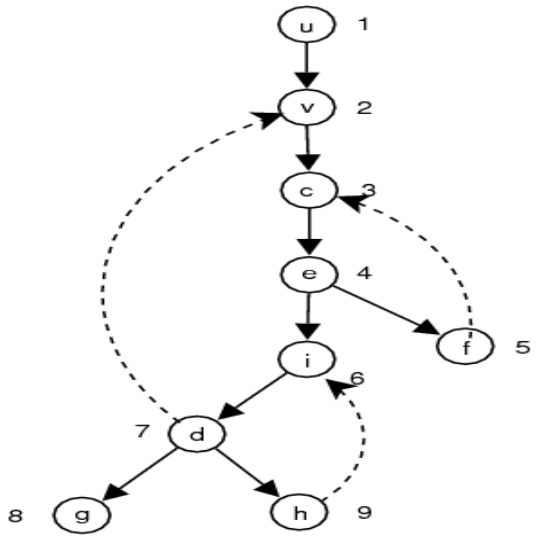


FIG. 5.5 – Arbre de parcours le long du lien $(u \longleftrightarrow v)$

Définition 1 *Un lien entre un nœud racine et un nœud voisin est dit critique si celui-ci est le seul chemin par lequel la racine peut atteindre ce nœud voisin.*

Un lien $(u \longleftrightarrow v)$ est dit lien critique :
si et seulement si $DFS(v) > 0$

3.4 Évaluation de la taille des partitions

Puisque l'algorithme DFS (donné ci-dessus) parcourt tous les nœuds accessibles à partir du nœud appelant (DFS(v)), il devient alors simple d'évaluer la taille des partitions. Nous disposons d'une variable *size* qui est mise à jour à chaque fois qu'un nœud est rencontré (ou visité) pour la première fois. Lorsqu'un nœud u différent de la racine est rencontré (ou visité) une seconde fois un indicateur s est positionné à *-1*, ceci correspond aux arcs de retour dans l'arbre résultant du parcours. Si c'est la racine de l'arbre qui est rencontré pour la première fois on sort de l'algorithme en retournant une grande valeur négative car le lien à statuer ne peut en aucun cas être un lien critique. Sur l'exemple de la figure 5.5 la valeur retournée par DFS(v) est 8 ce qui signifie que le lien $(u \longleftrightarrow v)$ est critique et la taille de la partition contenant v est 8. La proposition suivante résume les cas :

Propositions 1 *En utilisant l'arbre généré par le parcours DFS(.) :*

1. *Pour chaque arc* $(u \longrightarrow v)$ *si* $DFS(v) > 0$ *:*

(a) $(u \longleftrightarrow v)$ *est un lien critique*

(b) La taille de la partition contenant v *est DFS(v).*

2. *Si l'arête* $(u \longleftrightarrow v)$ *est un lien critique la taille de la partition contenant* u *est donné par :*

$$\sum_{w \text{ voisin de } u \text{ et } w \neq v} |DFS(w)|$$

3.5 Maintenir la connectivité

Une fois qu'un nœud a déterminé les liens critiques qui l'entoure, il doit en effet tenter d'éviter ou retarder la rupture de ces liens dans la mesure du possible. Avant cela nous allons d'abord évaluer le nombre maximal de liens critiques qu'il peut y avoir dans le voisinage d'un nœud quelconque u.

Définition 2 *Un lien critique* $(u \longleftrightarrow v)$ *est un segment* $[r - x, r]$ *de longueur* d *reliant* u *et* v*, où* r *le rayon de transmission et* x *une fraction ou portion de* r*.*

Pour trouver le nombre maximal de liens nous énonçons le lemme suivant

Lemme 5.1 *Un nœud racine* u *formant un lien critique avec un voisin* v *ne peut pas avoir en commun un voisin* w*.*

Preuve 5.1 *Supposons que* u *et* v *ont en commun un voisin* w*, en effet la racine* u *peut être visité (marqué) dans l'exécution DFS(v) lancée par* u *ce qui contredit la définition d'un lien critique donné en Définition 1.*

Théorème 5.1 *Le nombre maximal de liens critiques qu'il peut y avoir autour d'un nœud quelconque* u *est six.*

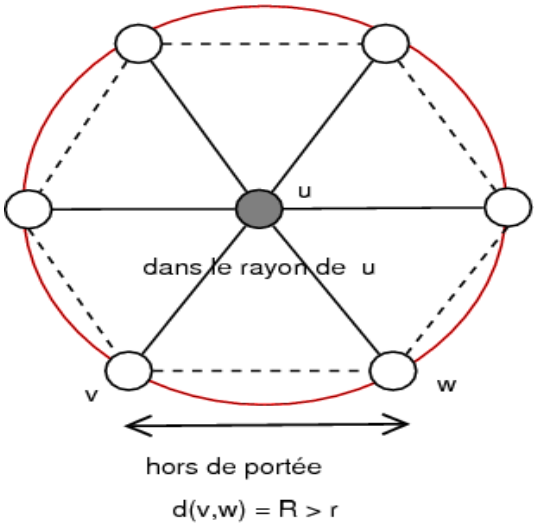


FIG. 5.6 – Un hexagone de coté R

Preuve 5.2 *La preuve peut s'obtenir géométriquement sur la figure 5.6. Le nombre de liens qui peuvent être formés autour du nœud racine sans que les nœuds impliqués par ces liens n'interagissent entre eux est six. Lorsqu'un septième nœud est ajouté (donc un septième lien) à cette configuration, il aura au moins deux voisins : la racine et un autre parmi les six. Ce qui contredit le Lemme 1.*

Nous proposons une maintenance de connectivité selon deux cas de figures :

Cas 1 : *Le nombre des liens critiques autour de u est égal à 1*

Pour illustrer notre démarche on va supposer que seul $(u \longleftrightarrow v)$ a été déclaré par u (racine) comme étant un lien critique. L'idée est de demander au nœud racine de se rapprocher du nœud v de t unités. La valeur de t est choisie de telle sorte que le lien en question puisse perdre son caractère critique. Si aucune vérification n'est faite, il est fort possible que cette manoeuvre entraîne dans le voisinage d'autres liens critiques. Avant d'entreprendre l'action la racine doit tester si aucun autre lien autour de lui ne sera affecté (avoir le caractère critique). Si le test échoue il renonce à la manoeuvre. Si le test réussi alors il va procéder comme suit :

Étape 1 : Établir un système de coordonnées temporaire (centré en u) par la méthode *GPS-Free* de Niculescu [3] pour les nœuds du sous-graphe G'_u. Les coordonnées de u seront dans ce cas $(0, 0)$ qu'on note tout de même par (x, y) et celles de v calculées par u seront notées par (x', y').

Étape 2 : Calculer ses nouvelles coordonnées (x_n, y_n) avec comme paramètre t par les équations suivantes :

$$x_n = x + t.cos\theta$$
$$y_n = y + t.sin\theta$$

et

$$\theta = tan^{-1}(\frac{y_{'} - y}{x_{'} - x})$$

Étape 3 : Se déplacer aux nouvelles coordonnées (x_n, y_n).

En rapprochant légèrement u de v comme le montre la figure 5.7, tout voisin w de v non loin de u (à une distance inférieure à r) aura grandes chances de devenir voisin de u. Cette technique permet d'éviter la rupture en deux ensembles connexes lorsque le lien $(u \longleftrightarrow v)$ se rompt.

Cas 2 : *Le nombre de liens critiques autour de u est supérieur à 1*

Dans cette situation il est difficile de trouver un compromis sans causer de rupture de liens. Mais on sait tout de même que le nombre le liens critiques n'excèdent pas six donc dans le pire des cas il y aura six partitions que le nœud racine tentera d'éviter ou retarder. Nous avons étudié le problème comme suit :

Soit u le nœud racine chaque voisin $v \in \Gamma(u)$ a fait l'objet d'un appel de la fonction $DFS(v)$ ce qui permet à u de connaître la taille des éventuelles partitions. Notons par $P = \{p_1, p_2 \dots p_i\}$ (avec $i \leq 6$) l'ensemble des partitions autour de u.

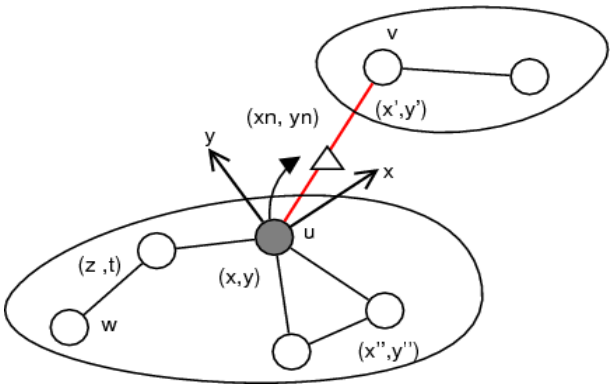


FIG. 5.7 – Déplacement de u vers v

On note par p^i la taille de chaque partition p_i. La racine évalue la moyenne m des p^i par $\frac{\sum p^i}{|P|}$.

• Si plus de la moitié des partitions a une taille supérieure ou égale à la moyenne m, la racine u veillera à ce que les données de chaque partition p_i soient copiées dans les autres partitions. L'idée est d'anticiper la formation de ces éventuelles partitions. Prenons par exemple p_i comme une partition source et p_j une partition destination. la racine est soit dans p_i ou soit dans p_j . Si u est dans la partition source son voisin disons v avec lequel il forme le lien critique est forcément dans la partition p_j. Le nœud v choisit (en fonction de la taille mémoire disponible) dans sa partition, des nœuds destinataires pouvant recevoir les données de la partition de u. La racine u collecte les données de sa partition et les achemine à v en utilisant le lien critique ($u \longleftrightarrow v$). A l'inverse la racine u choisit les nœuds potentiels dans sa partition et v copie les données de sa partition et les achemine à u par le lien ($u \longleftrightarrow v$).

• Si plus de la moitié des partitions ont une taille inférieure à la moyenne m, la racine donne l'avantage à la plus grande partition (la plus grande composante connexe). La racine u se déplace de t unités en direction de cette partition autrement dit vers le voisin v qui appartient à la plus grande partition. La racine établit un système de coordonnées temporaire (centré en lui-même) par la méthode *GPS-Free* de Niculescu [3] pour les seuls nœuds de son voisinage immédiat. Les coordonnées de u seront dans ce cas $(0, 0)$ qu'on note tout de même par (x, y) et celles de v calculées par u seront notées par (x', y'). La racine u calcule ses nouvelles coordonnées comme le (*Cas 1*).

La méthode décrite ci-dessus permet d'apporter des changements mineurs sur les trajectoires des nœuds impliqués dans les liens critiques sans créer d'incident au niveau de la topologie globale.

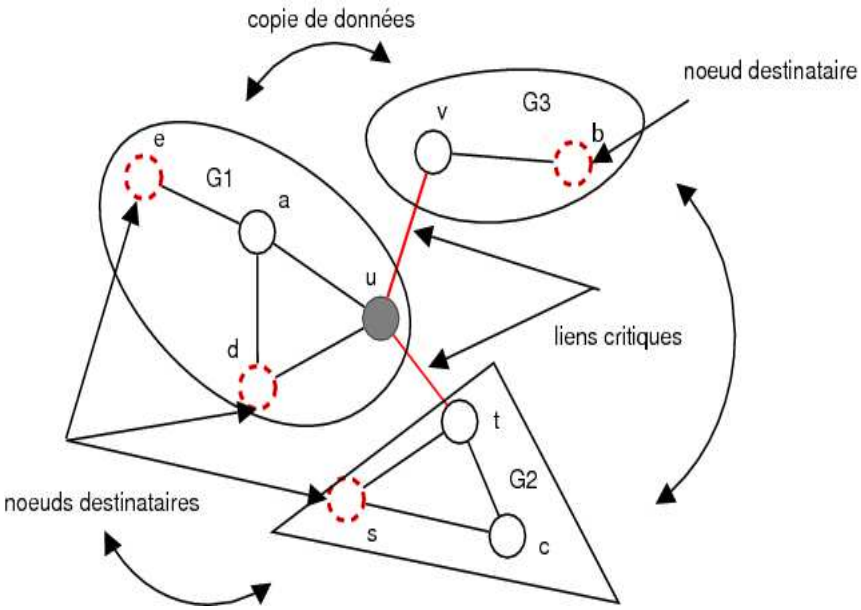


FIG. 5.8 – Copie de données de partition à partition

4 Tracking dans les réseaux de capteurs

Un réseau de capteurs est un réseau de type ad hoc dont les unités (terminaux) peuvent être statiques ou mobiles. Les terminaux disposent de trois fonctionnalités essentielles :

- La capacité de capturer des données relatives à l'environnement où ils sont physiquement placés, c'est à dire qu'un capteur peut mesurer des valeurs relatives à son environnement et les convertir en signaux électriques. Les données récupérées peuvent être de nature différente et la manière d'obtenir ces données est susceptible de varier.

- La capacité d'effectuer un traitement sur ces données récupérées,

- La capacité d'échanger ces données avec d'autres dispositifs.

On trouvera progressivement des capteurs partout, chargés de mesurer un grand nombre de signaux et de les communiquer via le réseau à un centre de contrôle (nœud puits ou sink). On peut les imaginer partout où la collecte d'information semble utile. Le tracking (suivie de cible) est l'une des applications les plus importantes de ces types de réseaux.

4.1 Préliminaires

On suppose une architecture clusterisée de k capteurs répartis dans une grille de taille NxM et d'un système de coordonnées centré au nœud sink(puits). Les k capteurs sont placés aux coordonnées respectives (x_1, y_1), (x_2, y_2), (x_3, y_3) ... (x_k, y_k). Dans cette architecture chaque clusterhead dispose de deux puissances de transmissions $p(r)$ et $P(R)$. On suppose qu'un capteur est capable de détecter une cible.

Un capteur code la présence d'une cible par 1 et si aune cible n'est présente cette information est codé par zéro (0).

La puissance de transmission $p(r)$ est utilisée pour une transmission en interne (transmission intra-cluster pour un rayon r) et la puissance $P(R)$ est réservée à la transmission entre clusterheads (pour un rayon R). Les clusterheads forment une dorsale ou backbone qui sert de support de communication entre le sink et l'ensemble des capteurs. La grille dans laquelle sont placés les k capteurs aux coordonnées (x_1, y_1), (x_2, y_2), (x_3, y_3) ... (x_k, y_k) est discrétisée ou découpée en des petits carrés de longueur égale à l'unité comme le montre la figure 5.9.

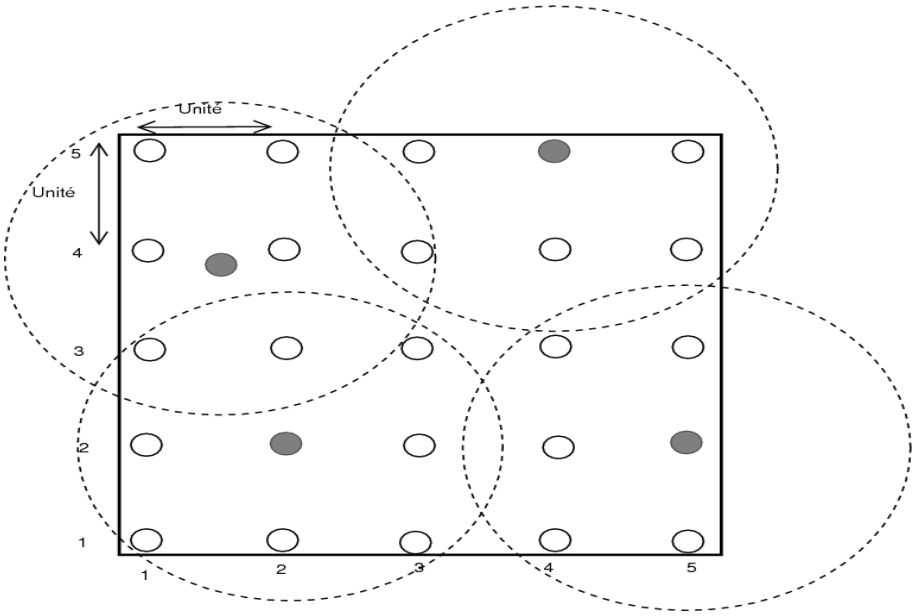


FIG. 5.9 – Grille de déploiement

Soit c_i le capteur déployé au point de coordonnées (x_i, y_i). Pour chaque point $P(x, y)$ de la grille nous évaluons par $d(c_i, P)$ la distance euclidienne entre le capteur c_i et ce point P. On note par $p_{xy}(c_i)$, la probabilité que le capteur c_i couvre (ou détecte) le point P de coordonnées (x, y). Cette probabilité de couverture est donnée par :

$$p_{xy}(c_i) = \begin{cases} 0 & \text{si } r + \tilde{r} \leq d(c_i, P) \\ 1 & \text{si } r - \tilde{r} \geq d(c_i, P) \\ expr(-\lambda x) & \text{sinon} \end{cases} \quad (5.1)$$

$\tilde{r}$ représente l'incertitude de mesure dans la détection, r le rayon de transmission et x la longueur du segment $[r - \tilde{r}, d(c_i, P)]$ et λ une constante donnée. La probabilité qu'un point $P(x, y)$ soit couvert ou détecté par deux capteurs c_i et c_j à la fois est exprimée par

$$p_{xy}(c_i, c_j) = 1 - (1 - p_{xy}(c_i))(1 - p_{xy}(c_j)) \quad (5.2)$$

4.2 Algorithme d'évaluation des probabilités de détection

Cet algorithme est exécuté une seule fois par le nœud sink, il permet d'évaluer l'étendue de la vue de chaque capteur. Il s'agit pour un capteur c_i donné et pour

chaque point P de la grille de mesurer la probabilité que ce point P soit couvert par le capteur c_i.

Algorithm 8 Évaluation_Couverture$(c_1, c_2, c_3, \ldots c_k)$

```
1: begin
2: Données : MAX_iteration
3: Variables : iter
4: iter := 0
5: while (iter < MAX_iteration) do
6:     for (chaque point P(x,y) avec x ∈ [1..M] , y ∈ [1..N]) do
7:         for (chaque capteur c_i ∈ {c_1, c_2, c_3..c_k}) do
8:             Calculer p_xy(c_i) en utilisant d(c_i, P) , r̃, r et λ
9:         end for
10:    end for
11:    iter := iter + 1
12: end while
```

4.3 Localisation d'une cible

L'architecture en clusters définit deux types de capteurs : les capteurs ordinaires et les clusterheads. Chaque capteur ordinaire est sous la responsabilité d'un seul clusterhead. Chaque capteur ordinaire prévient son clusterhead par un court message (`message de notification`) codé sur un bit de la présence ou non d'une cible. Il lui transmet un zéro (0) si aucune détection n'a eu lieu sinon un (1). Prenons par exemple un cluster noté C de clusterhead ω.

Soit m le nombre de capteurs ordinaires contenu dans le cluster C. Si un capteur ne détecte de cible dans la grille il transmet un zéro à son clusterhead. Si les m capteurs transmettent tous un zéro au clusterhead on obtient alors une chaîne ("000..0") de longueur m contenant que des zéros. Cette information reçue se traduit par le fait qu'aucun capteur n'a détecté de cible dans son rayon de transmission. Lorsque chaque capteur transmet un (1) au clusterhead alors on obtient également une chaîne ("111..1") de longueur m contenant que des uns (1). Cette information contrairement à la précédente signale la présence de cible détectée par tous les capteurs. Mais aussi une cible peut être détectée par un nombre n de capteurs sans être détecté par les $m-n$ capteurs restants. Dans ce cas la chaîne de longueur m qui sera transmise au clusterhead contiendra n un (1) et $m-n$ zéro (0). Énumérer l'ensemble des cas qui peuvent se présenter, revient à considérer l'ensemble des valeurs binaires possibles d'une chaîne de longueur m. Ce qui nous donne 2^m valeurs possibles.

Le clusterhead ω qui reçoit le message l'alerte, s'appuie sur la dorsale en utilisant sa puissance de transmission $P(R)$ pour prévenir à son tour le nœud sink. Les étapes qui résultent de cette alerte sont les suivantes :

- Le sink localise la cible et désigne en fonction d'une table de détection des points de la grille qu'il détient, les capteurs potentiels qui doivent être interrogés afin de fournir plus de détails sur la cible. Cette table dispose de $2^{|C|}$

entrées pour chaque cluster C de taille $|C|$.

- Il transmet ensuite une liste de capteurs potentiels au clusterhead ω.
- Le clusterhead ω enregistrera dans sa mémoire les informations utiles recueillies auprès des capteurs désignés dans son cluster en respectant l'ordre de priorité établie par le sink,
- Le clusterhead ω envoie enfin les informations recueillies au nœud sink.

4.4 Table des probabilités de détection

Comme nous venons de l'expliquer ci-dessus chaque clusterhead prévient le sink de la présence d'une cible (dans son cluster C) en lui transmettant une chaîne de longueur m avec m le nombre de capteurs ordinaires du cluster en question. Pour chaque possibilité (chaîne) $d_1 d_2 d_3 \dots d_m \in \{0, 1\}$ et pour chaque point $P(x, y)$ donné, nous évaluons la probabilité conditionnelle que le sink reçoit un message l'alertant de la présence d'une cible à ce point. Pour ce cluster C de m ($m < k$) capteurs ordinaires le sink disposera de $2^{|C|}$ entrées et pour chaque entrée on calcule la probabilité de détection par :

$$Pr_{xy}(i) = \prod_{c_i \in C} p_{xy}(c_i, i) \tag{5.3}$$

Le paramètre i est un entier tel que $0 \leq i \leq 2^{|C|}$ et la probabilité $p_{xy}(c_i, i) = p_{xy}(c_i)$ si c_i détecte une cible au point $P(x, y)$ sinon $p_{xy}(c_i, i) = 1 - p_{xy}(c_i)$. La table 5.2 illustre un exemple tiré de la figure 5.10 de taille 5 par 5 contenant 3 capteurs ordinaires appartenant au cluster C. Les paramètres λ, r, $\tilde{r}$ ont pour valeurs respectives 0.5, 2 et 1.

Algorithm 9 Probabilité_Détection($P(x, y)$,$(c_1, c_2, c_3, \dots c_k)$)

Déterminer S_{xy} l'ensemble des capteurs dans C qui peuvent détecter une cible au point P(x,y).

begin
for (chaque capteur $c_i \in \{c_1, c_2, c_3..c_k\}$) **do**
    if ($c_i \in C$ ET $d(c_i, P) \leq r + \tilde{r}$) **then**
        $S_{xy} = S_{xy} \cup \{c_i\}$
    end if
end for
for ($i = 0$, $0 \leq i < |S_{xy}|$) **do**
    if (c_i détecte $P(x, y)$) **then**
        $p_{xy}(c_i, i) := p_{xy}(c_i)$
    else
        $p_{xy}(c_i, i) := 1 - p_{xy}(c_i)$
    end if
    $Pr_{xy}(i) := \prod_{c_i \in C} p_{xy}(c_i, i)$
end for

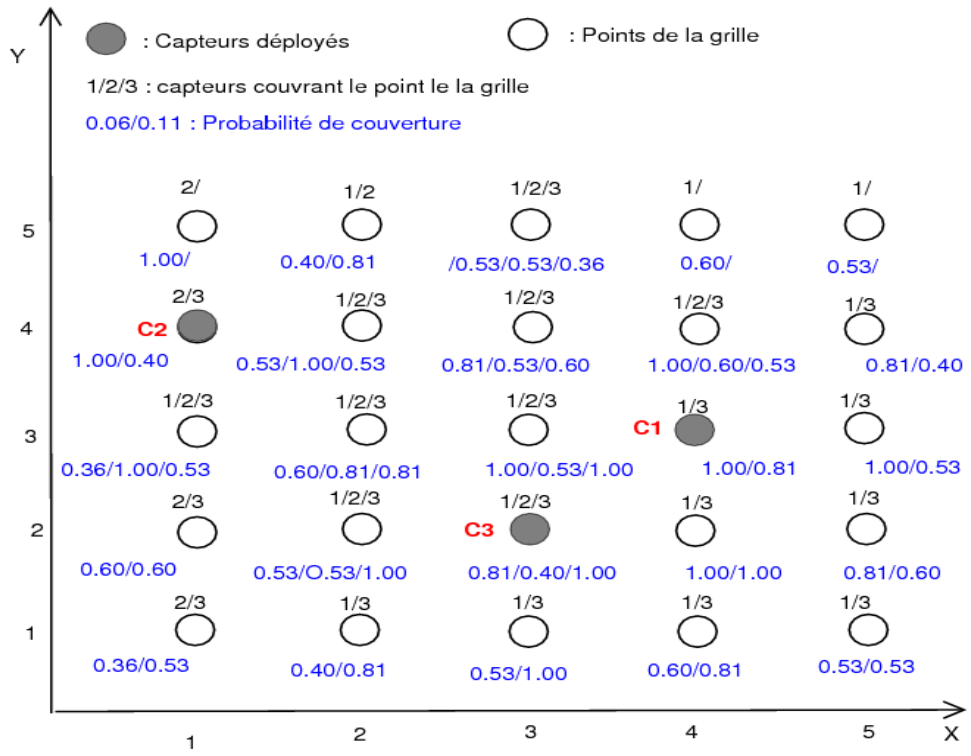


FIG. 5.10 – Probabilité de couverture des points de la grille

Considérons le point de coordonnées (2,4) de la figure 5.10 couvert par les capteurs c_1, c_2 et c_3 (codé par la chaîne binaire "111") de probabilité respective 0.53, 1.00 et 0.53. La probabilité conditionnelle associée à cette possibilité est donnée : $p_{24}(7) = p_{24}(c_1, 7)p_{24}(c_2, 7)p_{24}(c_3, 7) = 0.53\ .\ (1)\ .\ 0.53 = 0.28$. Les autres possibilités sont énumérées à la table 5.2.

indice	$d_1d_2d_3$	$Pr_{xy}(i)$	indice	$d_1d_2d_3$	$Pr_{xy}(i)$
0	000	0.00	4	100	0.00
1	001	0.00	5	101	0.00
2	010	0.2409	6	110	0.2491
3	011	0.2491	7	111	0.2709

TAB. 5.2 – Table des probabilités de détection pour le cluster C

4.5 Critère de sélection des capteurs potentiels

Après avoir déterminé la table des probabilités de détection pour chaque point de la grille, le sink procède à une localisation lorsqu'une cible est détectée par un ou plusieurs capteurs. Lorsque le sink reçoit d'un clusterhead ω une alerte de détection de cible, il utilise cette table des probabilités pour désigner les capteurs potentiels à interroger. Soit le cluster C contenant m capteurs, le sink dispose de 2^m entrées concernant ce clusterhead. Notons par $n_{xy}(t)$ le nombre de capteurs qui couvrent le point $P(x, y)$ et ayant signalés la présence d'une cible à ce point à l'instant t. On associe à ce point $P(x, y)$ un poids à l'instant t, $w_{xy}(t)$ défini par $w_{xy}(t) = \dfrac{n_{xy}(t)}{m}$.

On attribue à ce point un score à l'instant t, définie par :

$$SCORE_{xy}(indice(t)) = Pr_{xy}(indice(t)) \ . \ w_{xy}(t). \tag{5.4}$$

Les capteurs (du cluster en question) qui favorisent le plus grand score à ce point $P(x, y)$ sont retenus comme capteurs potentiels auprès desquels, le clusterhead ω doit obtenir plus d'information sur la cible. Prenons l'exemple de la figure 5.10 en supposant que les capteurs c_2, c_3 ont signalé à l'instant t la présence d'une cible au point $(2, 4)$. En utilisant la table des probabilités 5.2 on obtient les scores suivants : avec $w_{2,4}(t) = \frac{2}{3}$ et $m = 3$.

indice	$d_1d_2d_3$	$Pr_{xy}(i)$	$SCORE_{xy}(i(t))$
0	000	0.00	0.00
1	001	0.00	0.00
2	010	0.2409	0.1606
3	011	0.2491	0.1660
4	100	0.00	0.00
5	101	0.00	0.00
6	110	0.2491	0.1660
7	111	0.2709	0.1806

TAB. 5.3 – Évaluation des scores

Les résultats de cette table donne l'avantage au capteur c_2 puisque son score $SCORE_{24}(010) = 0.1606$ alors que celui de c_3, $SCORE_{24}(001) = 0.00$

5 Conclusion

Nous avons proposé en première partie de ce chapitre une approche qui vise à déterminer à l'avance les liens qui risquent de diviser tout réseau ad hoc connexe en plusieurs composantes (sous-ensembles) connexes. Prévoir les partitions peut être utile pour les applications dans un environnement ad hoc mobile. En effet, étant conscient d'un avenir de déconnexion du réseau, peut aider à assurer une meilleure qualité de service par l'adaptation du comportement des applications. Notre algorithme *DFS* permet à chaque nœud de détecter ces partitions dans son voisinage mais aussi dans la mesure du possible de les éviter ou de les retarder. La solution pour nous était de déplacer légèrement le nœud qui forme le lien critique (sans causer bien sûr d'autres liens critiques) vers des voisins immédiats, ce qui permet d'engendrer d'autres liens qui peuvent renforcer celui jugé critique. Mais cette solution ne peut être appliquée que lorsque le nombre de liens critiques est égal à 1. Dans le cas où il est supérieur à 1, une copie des données entre partitions s'opère.

Dans la seconde partie nous avons traité l'une des applications les plus importantes des réseaux des capteurs. Il s'agit bien évidemment de la détection et de la poursuite de cible. Nos capteurs forment un réseau clusterisé et sont répartis dans une grille discrétisée. Un algorithme centralisé au nœud puits permet de statuer sur les messages d'alerte qu'ils reçoivent. seuls les clusterheads transmettent des alertes

de détection (qui ont eu lieu dans le cluster) au nœud puits. Chaque clusterhead joue le rôle d'interface entre ses membres et le nœud sink. Les capteurs proches de la cible sont désignés par le sink et interrogés par le clusterhead. C'est une technique qui permet de reconstituer la trajectoire de la cible et sa vitesse moyenne de déplacement.

Chapitre 6

Collecte des données dans les réseaux de capteurs

1 Généralités

Les capteurs sont des petits dispositifs dont le but est de relever des informations concernant l'environnement dans lequel ils sont déployés et de les communiquer entre eux ou vers un nœud puits [22]. Les réseaux de capteurs sont en général constitués de centaines, voire de milliers, de nœuds et sont parfois conçus pour ne servir qu'une fois s'ils sont utilisés dans des milieux hostiles (champs de bataille, volcans, océans, etc.). Ces nœuds, de par leur taille, disposent de peu de ressources matérielles et surtout énergétiques. De ce fait, la découverte de route, la communication et la gestion de l'énergie doivent être gérées de façon à assurer une durée de vie maximale, tout en maintenant une fiabilité de communication suffisante. La figure 6.1 illustre un exemple d'installation de réseau de capteurs.

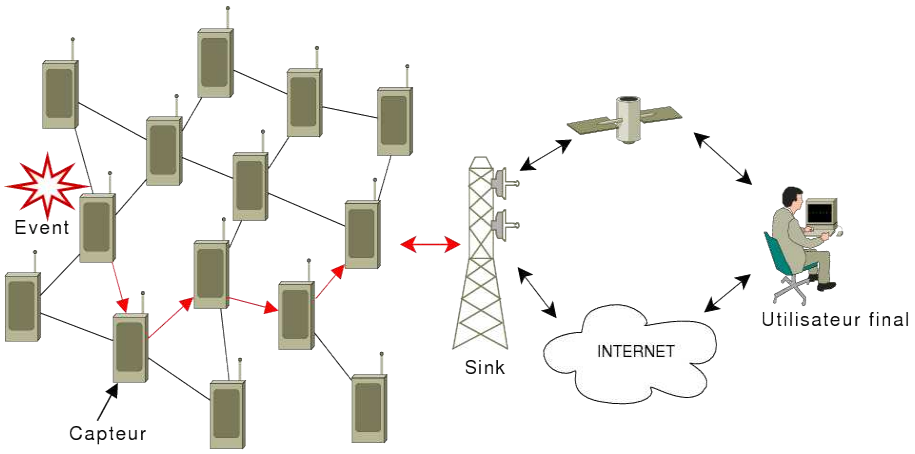


FIG. 6.1 – Un exemple de réseau de capteurs.

L'objectif fondamental d'un réseau de capteurs est de produire durant une période étendue des informations ou données locales brutes obtenues par les capteurs. Cet objectif doit être atteint dans un contexte de prolongement de la durée de vie utile du réseau et faire en sorte que le réseau demeure disponible et continue à fournir des informations précises en cas de défaillance matérielle. Le défi majeur auquel est

confronté la communauté de réseaux de capteurs est le développement de protocoles de communication ultralégers allant de la formation, de l'auto organisation, de la maintenance, de la collecte et de la fusion des données, du routage et bien d'autres encore [5, 28, 49]. Plusieurs techniques peuvent être utilisées pour l'interfaçage des réseaux de capteurs au monde extérieur et en particulier la collecte de l'information qu'ils produisent. Le plus simple consiste à utiliser un ou plusieurs nœuds spéciaux (puits ou sink en anglais) déployés aux côtés des capteurs. Dans ce scénario les données brutes recueillies par les capteurs sont transmises à un nœud[1] spécial (sink) qui se chargera d'acheminer ces données à l'utilisateur final. Mais dans certaines applications, il peut paraître impossible ou peu réaliste de déployer des nœuds spéciaux à l'intérieur du réseau de capteurs.

Dans de tels cas, les tâches de collecte d'information produite par le réseau de capteurs et celle de fournir une interface avec le monde extérieur peuvent être effectuées par des dispositifs (avion, hélicoptère) survolant la région de déploiement ou par une transmission laser à une constellation de satellites.

2 Localisation dans les réseaux de capteurs

Il a été reconnu que certaines applications nécessitent en plus des données brutes recueillies l'emplacement géographique des capteurs. Ceci a encouragé le développement de nouveaux protocoles de communication qui prennent en compte la localisation des capteurs. Après la phase de déploiement, les capteurs n'ont aucune idée de leur emplacement géographique ils doivent par conséquent être formés à se procurer cette information vitale. En plus de la limitation énergétique et du coût unitaire d'un équipement GPS il n'est pas envisageable de doter chaque capteur d'un tel équipement pour résoudre le problème de la localisation.

La localisation pour les différents capteurs consiste à déterminer les coordonnées géographiques de chaque capteur dans la zone de déploiement. Plusieurs solutions à ce problème sont fondées sur la multi-latération [32, 41, 49, 55]. La plupart de ces solutions exposées dans le chapitre 4 supposent l'existence de plusieurs points de références dont les positions sont connues d'avance. Les capteurs qui reçoivent des messages annonçant les positions d'au moins trois de ces points de référence peuvent à leur tour évaluer leur propre position.

3 Architecture Virtuelle de réseaux de capteurs

L'architecture virtuelle se compose d'un système de coordonnées dynamiques, d'une structure en grappes (appelés clusters en anglais) et d'un modèle de communication pour l'acheminement des données collectées par ces capteurs. Cette architecture peut servir de support pour un envoi de données en *Geocast* et pour une répartition en parallèle de plusieurs applications sur un même réseau de capteurs. Un cluster dans notre cas désigne un sous-ensemble de capteurs ayant les mêmes coordonnées. Il est important de noter qu'on l'obtient une fois le système de coordonnées établi.

[1] Le mot nœud désigne ici un capteur

3.1 Un modèle de capteur

Les capteurs sont des minuscules équipements déployés en masse dans une région quelconque et dépourvus d'identifiant. La seule source d'énergie dont ils disposent est leur batterie. Ils sont supposés opérationnels dés leur mise en service sans une intervention extérieure. Afin d'économiser la durée de vie du réseau qui dépend de l'énergie résiduelle, les capteurs sont en mode veille la plupart du temps, et se réveillent de façon aléatoire durant de courts intervalles de temps sous le contrôle d'une horloge interne.

Nous supposons que les capteurs sont équipés d'une radio à courte distance émetteur-récepteur. Les messages radios envoyés par un capteur ne peuvent atteindre que les capteurs dans sa proximité immédiate, une infime partie de l'ensemble de la population. A tout moment, un capteur, sera engagé dans l'exécution d'un ensemble fini d'opérations de base, ou sera en mode veille (endormi). Ces trois opérations de base sont la détection (collecter des mesures brutes), la fusion ou l'agrégation de données (obtenir des données cibles à partir des mesures brutes), le routage (communiquer les mesures brutes cibles, et les données de contrôle). Nous supposons que chaque opération exécutée par un capteur consomme une quantité d'énergie connue et fixe et un capteur en mode veille consomme une quantité d'énergie nulle.

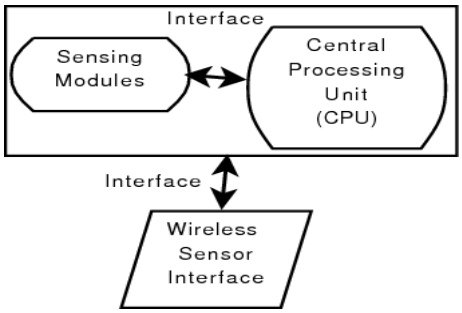


FIG. 6.2 – L'anatomie d'un capteur.

3.2 Structure d'un réseau de capteurs

Un réseau de capteurs se définit par un déploiement massif des capteurs par milliers voire des dizaines de milliers dans une région géographique quelconque. Le nombre de capteurs d'un tel réseau combiné aux caractéristiques de leur environnement (l'anonymat, budget énergétique limité, hostilité environnementale) constitue un défi pour les concepteurs de protocoles. Le contrôle de puissance de transmission est le premier moyen d'économie d'énergie du réseau. Par conséquent, le réseau doit être multi-sauts et seul un nombre limité de capteurs comptent le nœud sink (puits) dans leur voisinage. On suppose que la topologie du réseau n'est pas connue des capteurs et que la densité de déploiement fait que le réseau reste connexe. Notre modèle de réseau est composé d'un nœud sink et de plusieurs capteurs déployés aléatoirement dans son rayon de transmission.

3.3 Interfaçage avec le monde extérieur

Nous supposons que le réseau de capteurs est relié au monde extérieur via une passerelle appelée sink. La passerelle peut ne pas cohabiter avec les capteurs dans la même région. Sur la figure 6.3 l'interfaçage avec le monde extérieur est assurée par un hélicoptère survolant le réseau de capteurs en collectant les informations d'un groupe de nœuds appelés nœuds relais. Dans de tels scénarios la communication radio est utilisée entre les capteurs tandis qu'une communication laser est utilisée entre les nœuds relais et la passerelle ou sink. Notre étude porte sur la figure 6.4 avec une seule passerelle qui cohabite avec les capteurs dans la même région.

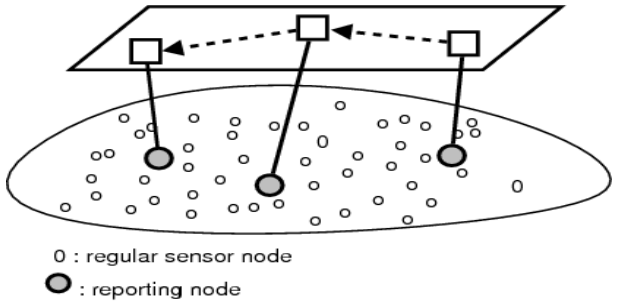


FIG. 6.3 – Un réseau de capteurs avec passerelle mobile et externe.

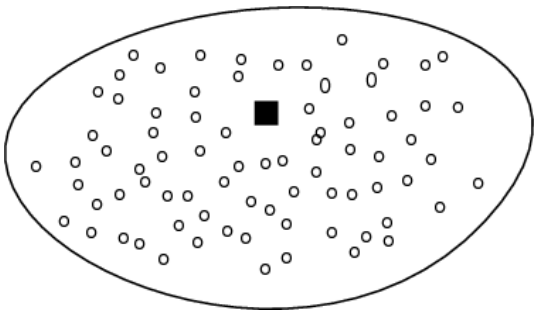


FIG. 6.4 – Un réseau de capteurs avec une passerelle statique et interne.

3.4 Modèle d'application des réseaux de capteurs

Le but de tout réseau de capteurs est d´ exécuter un certain nombre de tâches ou applications définies par son utilisateur. L'utilisateur n'étant pas directement relié au réseau dans la plupart des cas soumet ses tâches sous forme de requêtes au nœud sink afin d'atteindre le réseau. De cette description découle une hiérarchie de couches ou niveaux comme le montre la figure 6.5
Elle implique les couches suivantes :

- Couche application : niveau supérieur qui spécifie les tâches sous forme de requêtes d'une part et d'autre part consomme l'information produite par le réseau de capteurs.

- Couche Évènement (Middleware ou logiciel médiateur) : fournit une interface entre la couche réseau et la couche application, elle évalue les ressources du réseau avant de soumettre une tâche de niveau application.

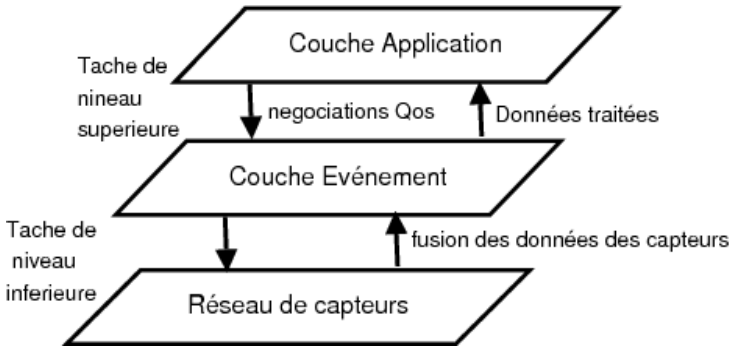


FIG. 6.5 – Modèle d'application et de gestion

Une tâche de niveau application est exprimée sous la forme (T, q) où T désigne la tâche en question et q la qualité de service souhaitée (figure 6.5). L'exécution de la tâche T avec la qualité de service q demandée dépend en grande partie des ressources du réseau telles que son énergie globale ou l'énergie résiduelle du cluster ou zone désignée pour la tâche T. Un exemple de tâche application peut être la détection du feu décrite par (Feu, q) ce qui se traduit par : Un feu est considéré dans la région lorsque la probabilité de détection est au moins égale à q.

Si les ressources ne permettent pas l'exécution de la tâche T avec la qualité de service q souhaitée alors la couche événement doit négocier une autre qualité de service $(q' < q)$ auprès de la couche application. Lorsque celle-ci est acquise la couche événement transforme ou réexprime la tâche (T, q') en une tâche primitive P avant de le soumettre à un ou plusieurs cluster(s) ou zone(s). Les zones désignées transmettent au sink les informations relatives à cette tâche en direction de l'utilisateur final. Afin de pouvoir désigner de façon unique un groupe de capteurs il faut construire par dessus du réseau une structure virtuelle avec un ensemble de fonctionnalités associées. On entend par structure l'organisation des capteurs pour faciliter leurs interactions. Les fonctionnalités consistent à maintenir cette structure et faciliter son utilisation. Cette structure est mise en place par un système de coordonnées dynamiques.

4 Système de coordonnées dynamiques

On suppose que le nœud sink peut effectuer k puissances trammissions omni-directionnelles et de m transmissons directionnelles. Les k puissances d'émission déterminent k cercles concentriques de rayon $r_1 < r_2 < r_3 \ldots < r_k$. Les k cercles concentriques définissent k couronnes. Les transmissons directionnelles quant à elles défissent m secteurs. La stratégie d'utilisation de ces transmissions permet de définir une structure composée de m.k groupes ou clusters. L'intersection d'une couronne i et secteur j définie un cluster de coordonnées (i, j). Les capteurs d'un même cluster ont les mêmes coordonnées. la figure 6.6 donne un exemple de structure (ou architecture) virtuelle.

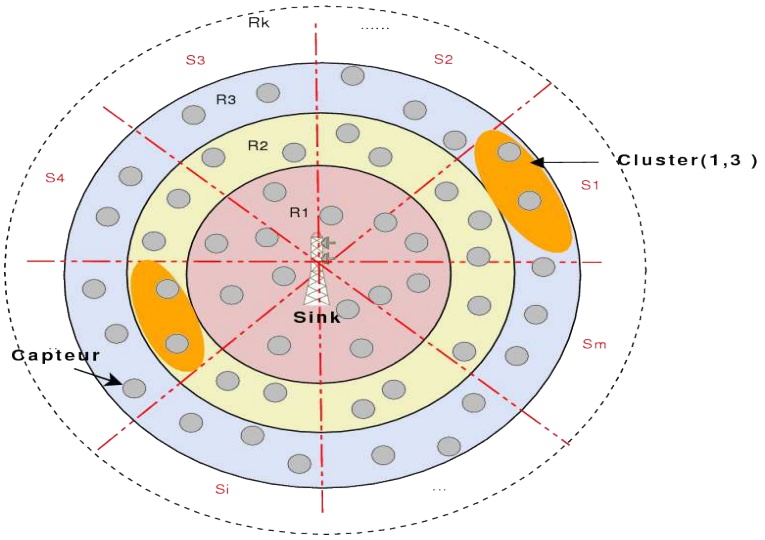


FIG. 6.6 – Structure virtuelle

4.1 Algorithme de formation des couronnes ou zones

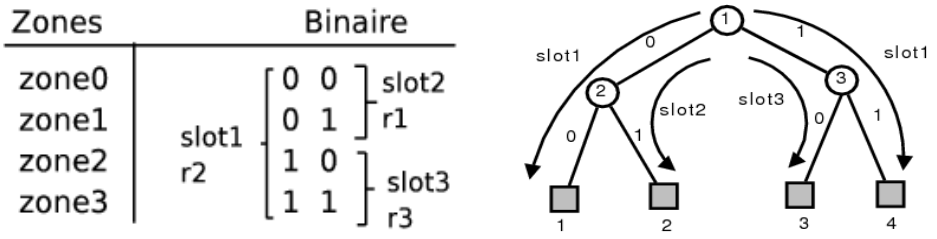
Soit k un entier[2] connus de tous les nœuds et soient k zones déterminées par les cercles concentriques de rayon $r_1 < r_2 < r_3 \ldots < r_k$ centré au nœud sink. L'idée est de permettre à chaque nœud du réseau de lire l'identité ou le numéro de la zone qui le contient. Pour cela chaque nœud lit une chaîne de $log(k)$ bits déterminant l´ identité de sa zone .

On suppose que le temps est découpé en des instants ou slots $s_1, s_2, \ldots, _{k-1}$ et on suppose que l'horloge maintenue en local par chaque nœud du réseau est synchronisée à celle du nœud sink. Pour former cette chaîne de longueur $log(k)$ bits $(b_1 b_2 .. b_k)$ on part du fait que la réception d'un signal émit par le nœud sink à chaque slot est codée par la valeur zéro (0) et la non réception par la valeur un (1). Pour économiser l'énergie globale du réseau chaque capteur est mis en veille dans la plupart du temps et se réveille pendant un cours intervalle de temps sous le contrôle de son horloge locale. On suppose que tous les nœuds sont éveillés durant le premier slot s_1. L'exemple ci-dessous permet de comprendre le principe pour une valeur donnée de k.

Exemple de formation de zones pour k =4

En partant de la représentation binaire des numéros des zones sur la figure 6.7, sur la colonne des bits de poids fort, seules les zones 0 et 1 ont leur bit de poids fort qui est à 0. Comme il a été dit précédemment la réception d'un signal par un nœud est codée par la valeur 0 et la non réception par la valeur 1. Par conséquent pour faire enregistrer la valeur 0 par les nœuds des zones 0 et 1 et la valeur 1 par les zones 2 et 3, le nœud sink doit alors utiliser ses rayons de transmissions r_1 et r_2.

[2]Nous supposons que k est une puissance de 2



(a) (b)

FIG. 6.7 – Formation des zones pour k=4

En effectuant un broadcast dans un rayon r_1 tous les nœuds de la *zone*0 recevront le message envoyé et mettent $b_1 = 0$. En effectuant un broadcast dans un rayon r_2 tous les nœuds de la *zone*1 recevront le message envoyé et mettent $b_1 = 0$. Étant donné que les numéros des zones sont consécutifs et que $r_1 < r_2$ alors le nœud sink effectuera un seul broadcast en utilisant son rayon de transmission r_2. Les nœuds des autres zones en occurrence 2 et 3 enregistrerons eux-mêmes la valeur 1 après l'expiration du délai du *slot*.

Durant le premier slot s_1 le nœud sink émet en broadcast dans un rayon r_2 .

Pour la colonne des bits de poids faible sur la figure 6.7, les bits à zéro ne sont pas consécutifs le premier bit à zéro apparaît dans la représentation binaire de la *zone*0. Le nœud sink émet en broadcast dans un rayon r_1 permettant aux nœuds de la *zone*0 de mettre leur bit $b_2 = 0$. Si le nœud sink se contente juste d'émettre dans un rayon r_1 tous les nœuds des autres zones enregistrerons pour b_2 la valeur 1 or seuls les nœuds de la *zone*1 ont besoin d'enregistrer leur dernier bit à 1. Pour empêcher ce phénomène l'idée est d'envoyer certains nœuds en mode veille et les réveiller au slot approprié afin qu'ils enregistrent la valeur souhaitée.

Durant le deuxième slot s_2 le nœud sink émet en broadcast dans un rayon r_1. Les nœuds des zones 2 et 3 seront en mode veille.

Le second bit à zéro dans la colonne C_2 apparaît dans l'écriture binaire de la *zone*2. Le nœud sink émet en broadcast dans un rayon r_3 permettant aux nœuds de la *zone*2 de mettre leur bit $b_2 = 0$.

Durant le troisième slot s_3 le nœud sink émet en broadcast dans un rayon r_3. Les nœuds des zones 0 et 1 seront à leur tour en mode veille.

4.2 Méthode analytique de détermination des zones

Considérons un arbre binaire complet à k feuilles noté A comme le montre la figure 6.8. Les feuilles de l'arbre sont numérotées de 1 à k. Pour chaque sommet l'arête menant à son sous arbre gauche est étiqueté par 0 celle menant au sous arbre droit est étiqueté par 1. Soit $l(1 \leq \ l \leq k)$ une feuille quelconque de A

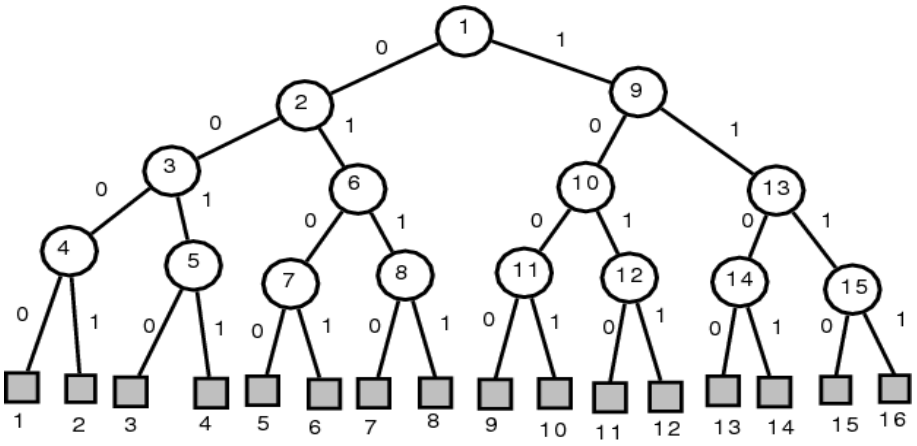


FIG. 6.8 – Étiquetage et parcours

et $b_1 b_2 \dots b_{log(k)}$ les étiquettes de l'unique chemin de la racine à l. Il est facile de prouver par un raisonnement simple que

$$l = 1 + \sum_{j=1}^{log(k)} b_j 2^{log(k)-j} \tag{6.1}$$

(En Appliquant l'équation (1) à la feuille numéro 7 nous avons : $7 = 1 + 0.2^3 + 1.2^2 + 1.2^1 + 0.2^0$). Soit A' l'arbre constitué essentiellement de nœuds intérieurs de A numérotés en préordre de 1 à $k-1$. Soit u un sommet quelconque dans A' et $b_1 b_2 \dots b_{log(i-1)}$ les étiquettes de l'unique chemin de la racine à u. Nous en déduisons le résultat suivant.

Lemme 6.1 *Soit $p(u)$ le rang de u dans un parcours préfixé de A' alors nous avons*

$$p(u) = 1 + \textstyle\sum_{j=1}^{i-1} c_j$$

$$avec\ c_j = \begin{cases} 1 & si\ b_j = 0 \\ \frac{k}{2^j} & si\ b_j = 1. \end{cases}$$

Preuve 6.1 *La preuve s'obtient par récurrence sur la profondeur i du sommet u dans A'. Pour $i = 1$ u désigne la racine de l'arbre donc $p(u) = 1$. Supposons que la relation reste vrai pour tout sommet $x/p(x) < p(u)$. Soit v le père de u dans A' et soit $b_1 b_2 \dots b_{(i-1)}$ les étiquettes de l'unique chemin qui mène de la racine à* u. *Alors u et v partagent le chemin $b_1 b_2 \dots b_{(i-2)}$ donc les $c_1 c_2 \dots c_{(i-2)}$. D'après l'hypothèse de récurrence,*

$$p(v) = 1 + \sum_{j=1}^{i-2} c_j. \tag{6.2}$$

Puisque v est le père de u alors on peut écrire

$$p(u) = p(v) + \begin{cases} 1 & si\ b_j = 0 \\ \frac{k}{2^i - 1} & si\ b_{i-1} = 1. \end{cases} \tag{6.3}$$

Si u est un fils gauche de v alors $b_{i-1} = 0$ et $c_{i-1} = 1$; sinon $b_{i-1} = 1$ et $c_{i-1} = \frac{k}{2i-1}$. En combinant les équations (2) et (3) on obtient

$$p(u) = 1 + \sum_{j=1}^{i-2} c_j + c_{i-1} = 1 + \sum_{j=1}^{i-1} c_j$$

Lemme 6.2 *Soit u un sommet quelconque de A' et $n(u)$ son rang dans le parcours infixe de A' . On note de 1 à k l'ordre dans lequel sont visités les sommets de A en utilisant le parcours infixe. Cette numérotation de 1 à k représente les feuilles de l'arbre A. Soit m le rang de la feuille de A le plus à droite dans le sous-arbre gauche enraciné en u alors $n(u) = m$.*

Preuve 6.2 *La preuve s'obtient par récurrence sur l'ordre de visite des sommets par un parcours infixe. Pour $n(u) = 1$, u est forcément la feuille la plus à gauche dans A' et son sous-arbre gauche dans A se résume à la feuille la plus à gauche dans A.*

Supposons que la relation reste vrai pour tout sommet x de A' tel que $n(x) < n(u)$. On distingue alors deux cas :

<u>Premier cas</u> : Soit v un parent de u dans A'. Notons par $A'(v)$ le sous arbre de A' enraciné en v. Le sommet u est dans ce cas la feuille la plus à gauche dans le sous-arbre droit de $A'(v)$. Soit q le rang de la feuille de A le plus à droite dans le sous-arbre gauche $A'(v)$. Par hypothèse de récurrence $n(v) = q$. Puisque u est une feuille dans A' il possède deux fils (feuilles) de rangs $q + 1$ et $q + 2$ dans A. Alors dans ce cas $n(u) = n(v) + 1 = q + 1$.

<u>Deuxième cas</u> : Soit u un parent de v dans A'. Notons par $A'(u)$ le sous-arbre de A' enraciné en u. Le sommet v est dans ce cas la feuille la plus à droite dans le sous-arbre gauche de $A'(u)$. Supposons que $n(u) = r$. Le sommet v dispose de deux fils (feuilles) dans A. Par hypothèse de récurrence ces fils sont aux rangs r et $r + 1$. Par conséquent $n(u) = n(v) + 1 = r + 1$.

Pour notre méthode de détermination des zones les paramètres $p(u)$ et $n(u)$ des sommets internes de A correspondent respectivement aux découpes du temps en des instants appelés slots et les rayons de transmissions utilisés par le nœud sink. Soit i un entier ($2 \leq i \leq log(k) - 1$) et supposons qu'un nœud u a renseigné son $i - 1$ bit de poids faible à la fin du slot s. Le résultat qui suit est un corollaire les lemmes (1) et (2).

Corollaire 6.1 *En ayant lus les bits $b_1 b_2 \dots b_{(i-1)}$ le capteur u doit se réveiller au slot $z = p(u) = 1 + \sum_{j=1}^{i-1} c_j$ pour lire son i^{eme} bit et durant ce slot z le nœud sink (passerelle) utilise un rayon de transmission $r_{n(z)}$.*

4.3 Formation des secteurs angulaires

Dans cette partie on subdivise la région de déploiement en m secteurs d'angles égaux à α. Comme le cas précédent chaque nœud doit lire une chaîne de longueur

$p(A)$	$n(A)$	Direction de l'antenne
1	2	π
2	1	$\frac{\pi}{2}$
3	3	$3\frac{\pi}{2}$

TAB. 6.1 – Rayons de tramsmission du sink

$log(m)$ pour déterminer l'identité du secteur angulaire qui le contient. Le temps est découpé en slot et à chaque slot correspond une antenne directionnelle d'angle β. Lorsque qu'un capteur u reçoit son $(i-1)^{eme}$ bit durant le slot s avec une antenne directionnelle d'angle β_{i-1} utilisée par le nœud sink alors u attendra le slot $z = p(u)$ pour lire son i^{eme} bit. Si $z \neq s+1$ le capteur u se met automatiquement en veille pour se réveiller au slot z sous le contrôle de son horloge locale. Le nombre de secteurs[3] angulaires est fixé par l'utilisateur.

Exemple de formation des secteurs angulaires pour m = 4

La figure 6.9 ci-dessous divise le plan en 4 parties d'angles $\alpha = \frac{\pi}{2}$. Les secteurs sont numérotés de 0 à $m-1$. Le paramètre $p(u)$ indique au capteur u le slot ou la date d'enregistrement de son prochain bit et le paramètre $n(u)$ intervient dans le calcul de l'angle (antenne directionnelle) sous lequel le nœud sink émettra durant ce slot. Les m secteurs représentent les feuilles de l'arbre binaire complet(voire figure 6.7).

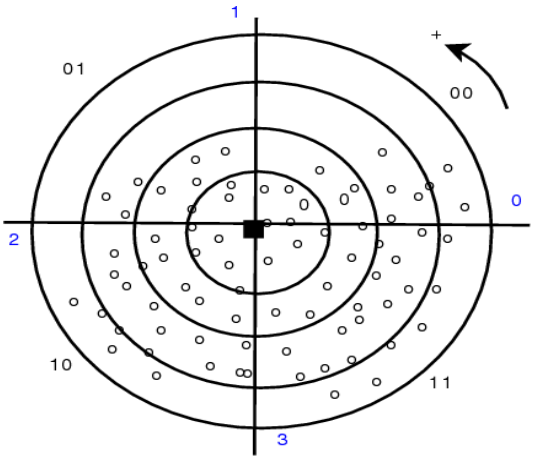


FIG. 6.9 – Formation des secteurs angulaires pour m=4

En partant de la représentation binaire des numéros des secteurs sur la figure 6.8 ; seuls les secteurs 0 et 1 ont leur bit de poids fort qui est à 0. La réception du signal émis par le nœud sink est codée par la valeur 0 et la non réception par la valeur 1. Par conséquent pour faire enregistrer la valeur 0 aux nœuds des secteurs 0 et 1 et la valeur 1 par les secteurs 2 et 3 le nœud sink doit alors utiliser son antenne directionnelle d'angle β_1. En effectuant un broadcast directionnel d'angle β_1 lors du slot s_1 tous les nœuds des secteurs 0 et 1 recevront le signal émis et mettront $b_1 = 0$.

[3]Le nombre de secteurs doivent être une puissance de 2

Les nœuds des autres secteurs, en l'occurrence 2 et 3 enregistrerons eux-mêmes la valeur 1 après l'expiration du délai du $slots_1$.

Durant le premier slot s_1 le nœud sink émet en broadcast directionnel d'angle $\beta_1 = \pi$.

Sur la colonne des bits de poids faible seuls le $secteur0$ et le $secteur2$ ont un zéro dans leur écriture binaire. Ces zéros ne sont pas consécutifs en effet les nœuds de ces secteurs ne peuvent pas enregistrer la même valeur dans le même slot. Les nœuds des secteurs 2 et 3 qui ont enregistré un 1 lors du précédent slot ($slot1$)seront endormis durant le présent slot ($slot2$). Le nœud sink émet en broadcast directionnel d'angle β_2 permettant aux nœuds du $secteur0$ de mettre leur bit b_2 à zéro ($b_2 = 0$) et ceux du $secteur1$ leur bit b_2 à 1 ($b_2 = 1$).

Durant le premier slot s_2 le nœud sink émet en broadcast directionnel d'angle $\beta_2 = \frac{\pi}{2}$.

Lors du dernier slot les nœuds des secteurs 2 et 3 seront réveillés et ceux des secteurs 0 et 1 auront déjà enregistré leur chaîne de longueur $log(m)$. Le nœud sink émet en broadcast directionnel d'angle β_3 permettant aux nœuds du $secteur2$ de mettre leur bit b_2 à zéro ($b_2 = 0$) et ceux du $secteur3$ leur bit b_2 à 1 ($b_2 = 1$).

Durant le troisième slot s_3 le nœud sink émet en broadcast directionnel d'angle $\beta_3 = 3\frac{\pi}{2}$.

4.4 Méthode analytique de détermination des secteurs

On part toujours de l'arbre binaire complet A mais cette fois-ci les feuilles représentent les m secteurs. On applique le parcours préfixé pour déterminer les différents slots $p(A)$. Le parcours infixe donne un second paramètre $n(A)$ qui permet d'évaluer les angles sous lesquels le nœud sink doit émettre. Sur l'arbre binaire complet de $m = 8$ feuilles on obtient la table ci-dessous :

Préfixe de A : $p(A)$	Infixe de A : $n(A)$	Direction de l'antenne
1	4	$n(u) * 2\pi/m = \pi$
2	2	$\pi/2$
3	1	$\pi/4$
4	3	$3\pi/4$
5	6	$3\pi/2$
6	5	$5\pi/4$
7	7	$7\pi/4$

TAB. 6.2 – Calcul des angles directionnels

Le réseau de capteurs résultant de la formation des zones et des secteurs angulaires centré au nœud sink est donné à la figure 6.10.

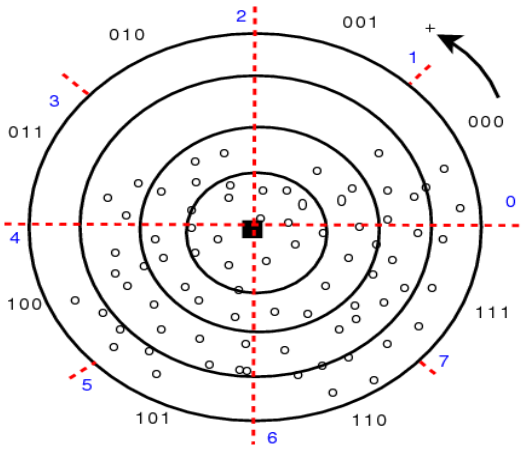


FIG. 6.10 – Formation des secteurs angulaires pour m=8

5 Structure en grappes ou clusters

La formation du système de coordonnées nous laisse un libre choix pour la désignation de notre structure en grappes (clusters). Ces clusters sont obtenus une fois après la formation des zones et des secteurs. On choisit comme cluster un ensemble de nœuds ayant les mêmes coordonnées c'est à dire appartenant à l'intersection d'une zone i et d'un secteur j noté $cluster(i, j)$. Cette méthode de clusterisation n'exige aucun échange de messages et aucune synchronisation des capteurs. Une application utilisateur (ou tâche) T est réexprimée sous une autre forme disons P avant d'être soumise à un ou plusieurs clusters par le biais du nœud sink.

5.1 Algorithme de découverte des clusters vides

Cette méthode de désignation des clusters ne permet pas au nœud sink de distinguer les clusters vides des clusters non vides. Cette partie de notre thèse propose un algorithme de détection des clusters non vides. En ayant une connaissance logique de la répartition des capteurs dans l'espace de déploiement, le sink peut déterminer à chaque instant un chemin entre vers un cluster donnée. On dénombre au total mk clusters représentés par une matrice noté $\hbar(k, m)$ que détient le nœud sink. Chaque nœud u du réseau dispose de coordonnées (x, y) désignant respectivement son numéro de zone et de secteur et on attribue au nœud sink les coordonnées *(-1,-1)* . Le temps est découpé en slots et chaque slot est de longueur τs. Les paramètres k, m, τ sont connus de tous les capteurs ainsi que les coordonnées du nœud sink. Les primitives de communications $radio.transmit(r, M)$ permet d'envoyer en broadcast le message M dans le rayon de transmission r et $radio.receive()$ réceptionne un message arrivant. On suppose que tous les capteurs sont synchronisés au nœud sink durant l'exécution de l'algorithme.

Table récapitulative des paramètres

- r : rayon de transmission courant du nœud sink
- $time()$: l'heure de l'horloge locale

- τ : durée d'un slot
- $msg(i,(x,y))$: message du sink émis dans la zone d'indice i avec les coordonnées (x,y) du nœud sink
- $msg(x,y)$: message d'un capteur avec (x,y) ses coordonnées
- $radio.transmit(r,M)$: transmission de M dans un rayon r
- $radio.receive()$: réception d'un message
- k : le nombre de zones (correspondant aux différents rayons de transmission du nœud sink)
- m : le nombre de secteurs (antennes directionnelles du nœud sink)
- $R(i)$: le i^{eme} rayon de transmission du nœud sink
- R : le rayon de transmission d'un capteur (capteurs homogènes)
- $\hbar(k,m)$: table des clusters initialement à zéro tenue par le sink
- L : liste des messages en attente

5.2 Algorithme du nœud sink

Avant l'exécution de l'algorithme le sink diffuse périodiquement un message d'alerte en précisant aux capteurs la date à laquelle doit débuter l'algorithme de découverte. Le nœud sink diffuse successivement le message M pour la découverte de clusters non vides dans chaque rayon de transmission $R(i)$ avec $i \geq k$. Pendant la durée du slot chaque réception de messages (en provenance d'un capteur voisin) fait objet d'une mise à jour de la table des clusters. Les messages émis par les capteurs en direction du sink sont principalement constitués des coordonnées de leurs émetteurs. A la fin de l'exécution de l'algorithme un $cluster(i,j)$ est considéré vide lorsque $\hbar(i,j) = 0$.

Algorithm 10 Découverte_cluster_vide(k)

Require: $k \in \aleph; k > 0; M; N; duree$

```
1:  begin
2:  for i := 0 to k − 1 do
3:      r := R(i + 1)
4:      M := msg(i, (" − 1", " − 1"))
5:      radio.transmit(r, M)
6:      duree := time() + τ
7:      while (time() < duree) do
8:          if (N = radio.receive()) = true) then
9:              x = N.x
10:             y = N.y
11:             if Tab(x, y) = 0 then
12:                 ħ(x, y) = 1
13:             end if
14:         end if
15:     end while
16: end for
17: end
```

5.3 Algorithme d'un capteur

Chaque capteur qui reçoit le message d'alerte durant sa période de réveil se prépare à l'exécution de l'algorithme en commençant par calculer la durée nécessaire voulue par le nœud sink.

Réception en provenance du nœud sink : un nœud u qui reçoit ce message pendant cette durée vérifie s'il se trouve dans le rayon de transmission spécifié dans M si c'est le cas il construit son message $msg(x, y)$ avec (x, y) ses propres coordonnées et le diffuse en direction du nœud sink. Si le nœud u ne se trouve pas dans le rayon de transmision spécifié dans M alors il se prépare à relayer tout $msg(x, y)$ qu'il recevra.

Réception en provenance d'un nœud voisin : le nœud v qui reçoit un message d'un voisin u le relaye en direction du sink ainsi que tous les messages en attente s'il a au préalable reçu le message du nœud sink sinon il ajoute le nouveau $msg(x, y)$ à sa liste d'attente.

Algorithm 11 Découverte_cluster_vide(k)

Require: $k \in \aleph; k > 0; M; N; duree; L$

```
1: begin
2: duree := time() + τ
3: while (time() < duree) do
4:     if (M = radio.receive()) = true) then
5:         if (N.i = x) then
6:             N := msg(x, y)
7:             radio.transmit(R, N)
8:             if (nonvide(L)) then
9:                 for chaque l ∈ L do
10:                    radio.transmit(R, l.N)
11:                end for
12:            end if
13:        else
14:            if (N = radio.receive()) = true) then
15:                if (n.i < x) then
16:                    radio.transmit(R, N)
17:                end if
18:            end if
19:        end if
20:    else
21:        if (N = radio.receive()) = true) then
22:            L := L ∪ {N}
23:        end if
24:    end if
25: end while
26: end
```

L'algorithme de découverte des clusters vides vise à donner au nœud sink une vue globale et statique du réseau. Cette vue globale est représentée dans une matrice

$\hbar$ de taille $k.m$ avec k le nombre de zones et m le nombre de secteurs. La figure 6.11 donne une illustration pour $k = 4$ et $m = 8$. Puisque les secteurs sont numérotes (en ordre croissant) dans le sens trigonométriques cela établit une adjacence entre les secteurs. Cette adjacence permet pour un cluster donné de trouver un chemin qui mène au nœud sink. Une ligne dans cette table dont toutes valeurs sont à 1 signifie qu'il existe un chemin entre n'importe quel cluster et le sink. On parle de modèle de communication centralisé. Lorsqu'il n'existe pas de chemin dans le même secteur on parle de modèle communication distribué. Le choix du modèle de communication est présenté au paragraphe suivant.

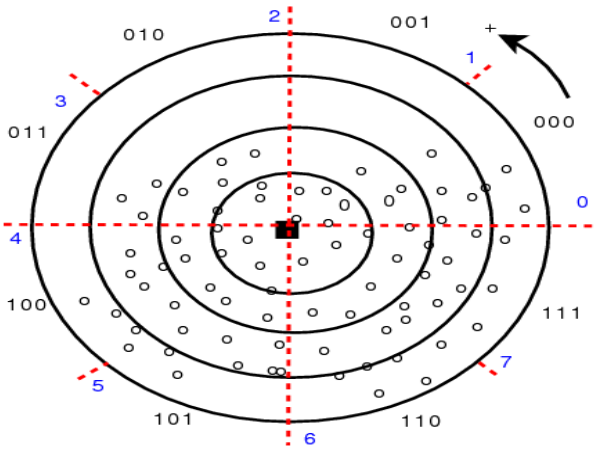


FIG. 6.11 – Architecture virtuelle

Secteurs	Zones			
	R_1	R_2	R_3	R_4
S_0	1	1	1	1
S_1	1	0	0	0
S_3	1	1	0	0
S_4	1	1	1	0
S_5	1	1	1	1
S_6	1	1	1	1
S_7	1	1	1	1

TAB. 6.3 – Table des clusters correspondant à la figure 6.11

5.4 Description et routage des données

Dépendant de la façon dont les données sont transmises au nœud sink nous proposons deux modèles de communication. Un modèle de communication centralisé présenté à la figure 6.11 dans lequel les données collectées dans un cluster source $cluster(i, j)$ sont routées au sein du même secteur angulaire j en direction du nœud sink. Un modèle de communication distribué présenté à la figure 6.11 intervient lorsque le modèle de communication centralisé n'est pas directement applicable. Ce cas peut se présenter lorsqu'il n'existe pas de chemin interne (au sein du même secteur) pouvant relier le cluster source au nœud sink. Les données collectées sont alors routées de secteur à secteur jusqu'au secteur pouvant atteindre le nœud sink

par un chemin interne.

Notons par T la tâche utilisateur (niveau application) qui doit être soumise au réseau de capteurs avec une qualité de service q. Un exemple de tâche de niveau application peut être une détection de feu, de gaz, de chaleur ou autre dans une partie de la région de déploiement. Dans notre exemple on suppose que T désigne une détection de feu qu'on note par $T = (Feu, q)$. On considère qu'il a vraiment feu lorsque la fumée détectée excède une certaine valeur q donnée. Le logiciel médiateur s'exécutant sur le nœud sink, réexprime la tâche T en une autre tâche noté P identifiant ainsi un cluster source pour l'exécution de la tâche et un modèle de communication pour l'acheminement des données collectées. La tâche P est un 5-uplets qui se présente comme suit : (A, S, D, π, q)

- A : est la dénomination de la tâche décrite par l'utilisateur (feu par exemple)
- S : identité du cluster désigné pour son exécution
- D : l'identité du nœud sink
- π : spécifie la séquence de clusters relais pour acheminer les données de S à D
- q : la qualité de service souhaitée par l'utilisateur.

Le choix des clusters relais s'effectue par le nœud sink en consultant la table des clusters qu'il détient. Supposons deux tâches $T1 : (Feu, q)$ et $T2 : (Gaz, q')$ de niveau application qui doivent être soumises au réseau. En consultant la table des clusters relative au réseau, le nœud sink peut réexprimer les tâches $T1 : (Feu, q)$ et $T2 : (Gaz, q')$ en deux tâches $P1$ et $P2$ de la manière suivante :

$$P1 : (Feu, S1, Sink, \{(2,0), (2,7), (2,6), (1,6), (0,6), \}, q)$$

$$P2 : (Gaz, S2, Sink, \{(2,5), (1,5), (0,5)\}, q')$$

Les informations ainsi collectées doivent être routées de cluster en cluster jusqu'au nœud sink selon le modèle de communication choisi.

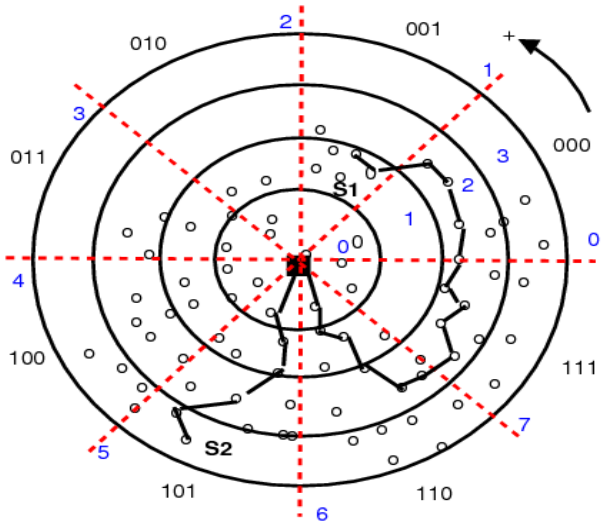


FIG. 6.12 – Illustration du routage des données

5.5 Fusion des données

Une fois que les données sont collectées par un ensemble de capteurs désignés, la seconde étape consisterait à fusionner ces données afin de minimiser le coût du trafic en direction du nœud sink. Notons par (i, j) l'identité du cluster choisi par le nœud sink pour l'exécution de la tâche T. Soient $\{(i-1, j), (i-2, j)....(1, j)\}$ la séquence des clusters relais désignés pour le routage des données dans le cas d'une communication centralisée. Il reste à savoir comment la tâche T est -elle soumise ? Le nœud sink commence par lancer un appel d'offre pendant un intervalle de temps Δ choisi en fonction de la durée de mise en veille périodique des capteurs. Le sink désigne dans l'appel d'offre les clusters qui doivent être impliqués pour la tâche T. Les capteurs en état de réveil au moment de l'appel, resteront réveillés après avoir notifié le message du nœud sink. Chaque capteur impliqué se préparera alors à l'exécution de la tâche T.

Soit χ l'ensemble de capteurs relais ou désignés et Ω l'ensemble des capteurs sources du cluster (i, j). On part de l'hypothèse que les capteurs qui jouent le rôle de routeurs sont synchronisés. En supposant que les données relatives à la tâche T soient une puissance de deux, on pourra alors les répartir en 2^d groupes disjoints. Chaque groupe ou capteur $u \in \Omega$ aura une chaîne de d bits à transmettre. Partant de ces hypothèses nous proposons deux approches de fusion des données.

La première approche consiste à fusionner toutes les données individuelles en une seule donnée qui sera ensuite transmise au nœud sink. L'opération de fusion est étroitement liée à la description de la tâche en soumission. En s'appuyant sur l'algorithme de Nakano et Olariu [42, 43] on attribue une identité temporaire allant de 1 à $|\Omega|$ à chaque capteur de l'ensemble Ω. En utilisant leur identité temporaire chaque capteur transmettra ses données individuelles au capteur dont l'identité est la plus petite(capteur d'identifiant égale à 1). Ce dernier se chargera de la fusion et de l'envoi du résultat final (résultat de la fusion) au nœud sink.

L'avantage de cette approche est qu' aucune donnée ne sera perdue d'une part et d'autre part elles seront correctement fusionnées. Il existe cependant deux inconvénients : le premier inconvénient est l'initialisation de l'algorithme de Nakano et Olariu [42, 43], celui-ci exige à chaque capteur de l'ensemble Ω une quantité d'énergie supplémentaire proportionnelle à la taille de Ω. Le deuxième inconvénient est l'approche qui consiste à tout centraliser au sein d'une seule et même entité. La défaillance de ce nœud central peut provoquer la perte des données et l'utilisation abusive d'un nœud central réduira considérablement la durée de vie de sa batterie.

La seconde approche quant à elle n'exige aucune attribution d'identité aux capteurs du cluster source (i, j). L'idée est la suivante : chaque capteur $u \in \Omega$ transmet les données collectées bit par bit en commençant de la gauche vers la droite. On découpe le temps en slots de durée égale et chaque capteur transmet un bit par slot. Pour alléger le trafic un bit de valeur 0 ne sera pas transmis tandis qu'un bit de valeur 1 sera transmis. Les capteurs du cluster $(i-1, j)$ désignés pour jouer le rôle de capteurs relais réceptionnent les données envoyées de la manière suivante :

- Aucun bit reçu durant le slot $\longrightarrow$ enregistrement d'un 0 (zéro)
- Un bit de valeur 1 reçu durant le slot$\longrightarrow$ enregistrement de 1
- Collision détectée $\longrightarrow$ enregistrement de 1

Les capteurs relais du cluster $(i-1, j)$ ne peuvent relayer aux capteurs du cluster $(i-2, j)$ que lorsque qu'ils auront enregistré une chaîne de longueur d. Cette façon de fusionner les données revient à effectuer une opération logique sur les valeurs reçues. Au final les capteurs relais auront tous comme données à transmettre le résultat de l'opération binaire effectuée sur les données reçues.

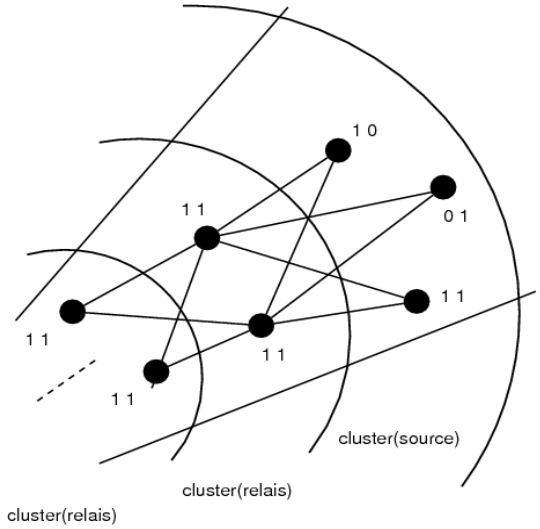


FIG. 6.13 – Illustration de la fusion des données

La fusion des données proposées ci-dessus est étroitement liée à la tâche utilisateur et de sa description. Prenons deux exemples concrets pour mettre en oeuvre cette fusion des données. On note par T_1 la tâche qui constitue à mesurer l'aridité d'un sol dans un domaine agricole et T_2 la poursuite de cible. Le domaine agricole se base sur l'indice d'aridité pour distinguer les régions arides des régions humides afin de déclencher un système d'arrosage automatique. Le tableau ci-dessous donne une description de la tâche T_1.

Incide d'aridité	$[0.0, 0.3[$	$[0.3, 0.2[$	$[0.2, 0.5[$	> 0.5
code	11	10	01	00

TAB. 6.4 – Description et Codage de la tâche T_1

Dans l'exemple du système d'arrosage d'un domaine agricole (voir tableau), on peut donner comme situaion d'urgence (il faut arroser) si et seulement si le code associé à l'indice d'aridité a son bit de poids fort égal à 1. La fusion (opération binaire) des codes décrivant une situation d'urgence doit produire un code qui exprimera toujours la situation d'urgence. On peut dans ce cas utiliser comme opération binaire le **OU** logique entre un ou plusieurs situations d'urgence. Le résultat de cette opération donnera toujours un code qui décrit une situation d'urgence.

La tâche T_2 cherche à produire la trace d'une cible en mouvement. Soumise à tous les clusters (réseau entier) il peut permettre de relever à des instants les positions de la cible en mouvement. Il revient à chaque capteur ayant détecté la cible d'envoyer ses propres coordonnées $(zone, secteur)$ en direction du nœud sink. Dans ce cas de figure le résultat de la fusion des codes associés aux coordonnées doit rester inchanger. On peut utiliser de nouveau le **OU** logique comme opérateur binaire. Le table 6.5 donne une description de la tâche T_2.

Cluster	$(0,0)$	$(0,1)$	$(0,1)$	$(0,3)$	...	(i,j)	...	$(7,3)$
code	00000	00001	00010	00011	...	bin(i)bin(j)	...	11111

TAB. 6.5 – Description et Codage de la tâche T_2

6 Conclusion

Nous avons proposé dans ce chapitre un modèle virtuel de réseau de capteurs anonymes. les capteurs étant des dispositifs de capacités très limités il devient alors indispensable de trouver une approche permettant d'organiser ou de faire collaborer à moindre coût ces capteurs dans un but bien précis. Le nœud sink est le seul équipement qui dispose d'assez d'énergie et de puissance de transmission pouvant atteindre l'ensemble des capteurs repartis dans la région d'intérêt. Une stratégie de gestion de ou des antennes du nœud sink nous a permis de proposer une méthode de localisation sans inonder le réseau par échange de messages d'information de position. Il nous a également permis d'esquisser une structure en grappes offrant un support de routage simple et efficace mais aussi pratique pour fusion des données .

Cette structure permet des paralléliser plusieurs applications utilisateur. Les données ainsi recueillies peuvent être acheminées selon un modèle de communication centralisé ou distribué défini à l'avance par le sink. Nous avons également proposé une approche de fusion des données collectées afin de minimiser le trafic sous-jacent. Cette fusion des données dépend de la description de application en question. Quelque soit l'application l'idée consiste à compacter des informations reçues des capteurs avant de les transmettre au nœud sink de sorte que celles-ci gardent leur pertinence. Le nœud sink recevant l'information finale réagira en effet selon l'importance du message. Mais sur ce modèle, la synchronisation des capteurs à l'horloge globale du sink après leurs périodes de sommeil reste à faire.

Chapitre 7

Conclusion et Perspectives

1 Conclusion

1.1 Partitionnement

Nous avons présenté au chapitre 2 différentes techniques de partitionnement. Le point de divergence principal entre elles est l'heuristique utilisée pour déterminer quel nœud sera le *clusterhead*. En effet, certaines se basent sur les identifiants des nœuds, d'autres sur leur degré, tandis que Basagni lui préfère utiliser un système de poids générique,liée à la mobilité. Très clairement, ce système a l'avantage d'être plus proche de la réalité du réseau que les autres solutions. Le problème est désormais de formuler un bon système de poids permettant d'installer une structure de clusters stable. Le système de poids choisi doit dans la mesure du possible refléter l'état du réseau. L'état du réseau est sa simple description à des instants donnés par un certain nombre de paramètres qui peuvent être liés à la mobilité, au trafic, à l'énergie résiduelle, à la densité etc...

Notre algorithme de partitionnement sans unicité de poids prend en compte les paramètres suivants : vitesse moyenne de déplacement des nœuds, l'énergie résiduelle et la densité. Nous avons également comparé notre heuristique des solutions existantes en terme de nombre de message générés durant le processus de partitionnement. En effet, disposer d'une structure de clusters fiable permet d'améliorer très nettement l'implémentation de nombreux protocoles de routage dédiés aux réseaux mobiles ad hoc. Un routage hiérarchique semble être un bon candidat.

1.2 Localisation et Geocasting

Nous avons proposé également dans cette thèse un algorithme d'estimation de position *L-Libre* permettant à des nœuds simples d'acquérir une position dans un système de coordonnées centré au nœud *sink* avec quatre ancres fictifs. Donner une position à des nœuds qui en sont normalement dépourvus n'est pas une tâche simple. L'intérêt est de permettre à ces nœuds simples d'être membres d'un réseau hétérogène. Notre algorithme `L-Libre` comporte trois étapes : la première consiste à faire parvenir à l'ensemble des nœuds les positions et les distances des ancres y compris le nœud sink. La seconde étape consiste pour chaque nœud de déterminer si les ancres sont des voisins immédiats si tel est cas les distances réelles seront

utilisées sinon les distances estimées seront utilisées. Chaque nœud doit également lever l'ambiguïté lorsque deux positions sont possibles. Enfin la dernière étape consiste à estimer la position. Des méthode mathématiques de résolution d'équations permettent l'estimation de position selon la situation.

On a très souvent entendu parler des modes de transmissions unicast (de un vers un), de multicast (de un vers plusieurs) et broadcast (de un vers tous). Il existe un autre mode de transmission appelé geocast qui consiste à transmettre des données à un groupe de nœuds situés dans une région géographique spécifiée par l'émetteur. Nous avons proposé une approche du geocasting avec garantie de livraison à la destination. Notre algorithme s'inspire du traditionnel algorithme *DSR*. Nous avons rajouté à ce protocole deux autres messages (`Position_Echange_msg`, `Impass_msg` pour aider à la prise de décision lors d'un `échec de route` dans un domaine restreint (forwarding group δ) calculé à l'avance par le nœud émetteur. Le message de type `Position_Echange_msg` permet de collecter les informations de position du voisinage et le message de type `Impass_msg` permet de signaler un `échec de route` à un nœud prédécesseur. Lorsqu'un échec de route est signalé par tous les voisins directs du nœud émetteur celui-ci doit réadapter le `forwarding group` pour procéder à une nouvelle découverte de routes.

1.3 Maintenance de la connectivité et le tracking

Notre approche sur la maintenance de connectivité vise à déterminer à l'avance les liens qui risquent de scinder tout réseau ad hoc connexe en plusieurs sous-ensembles connexes on parle dans ce cas de partitionnement ou décomposition. Prévoir les partitions peut être utile pour les applications dans un environnement ad hoc mobile. En effet, étant conscient d'un avenir de déconnexion du réseau, peut aider à assurer une meilleure qualité de service par l'adaptation du comportemet des applications. Notre algorithme *DFS* permet à chaque nœud de détecter ces partitions dans son voisinage mais aussi dans la mesure du possible de les éviter ou de les retarder. La solution pour nous était de déplacer légèrement le nœud qui forme le lien critique (sans causer bien sûr d'autres liens critiques) vers des voisins immédiats, ce qui permet d'engendrer d'autres liens qui peuvent renforcer celui jugé critique. Mais cette solution ne peut être appliquée que lorsque le nombre de lien critique est égal à 1. Dans le cas où il est supérieur à 1, une copie des données entre partitions s'opère.

Dans le même chapitre nous avons présenté l'une des applications les importantes des réseaux de capteurs . Il s'agit bien évidemment du tracking ou détection et poursuite de cible. Nos capteurs forment un réseau clusterisé et sont répartis dans une grille. Des algorithmes centralisés au nœud puits permet de statuer sur les messages d'alerte qu'ils reçoivent. seuls les clusterheads transmettent des alertes de détection au nœud puits. Chaque clusterhead joue le rôle d'interface entre ses membres et le nœud sink. Les capteurs proches de la cible sont désignés par le sink et interrogés par le clusterhead. C'est une technique qui permet de reconstituer la trajectoire de la cible et sa vitesse moyenne de déplacement.

1.4 Architecture virtuelle et fusion des données

Et enfin notre architecture virtuelle de réseau de capteurs anonymes. Les capteurs étant des dispositifs de capacités très limitées il devient alors indispensable de trouver une approche permettant d'organiser ou de faire collaborer à moindre coût ces capteurs dans un but bien précis. Le nœud sink est le seul équipement qui dispose d'assez d'énergie et de puissance de transmission pouvant atteindre l'ensemble des capteurs repartis dans la région d'intérêt. Une stratégie de gestion de ou des antennes du nœud sink nous a permis de proposer une méthode localisation sans inonder le réseau par échange de messages d'informations de positions. Il nous a également permis d'esquisser une structure en grappes offrant un support de routage simple et efficace. Les données ainsi recueillies par les capteurs peuvent être acheminées selon un modèle de communication centralisé ou distribué défini à l'avance par le sink. Nous avons également proposé une approche de fusion des données collectées afin de minimiser le trafic sous-jacent. Cette fusion des données dépend de la description de application en question. Quelque soit l'application l'idée consiste à compacter des informations reçues des capteurs avant des les transmettre au nœud sink de sorte que celles-ci gardent leur pertinence.

2 Perspectives

Il reste toutefois bien des voies à explorer pour améliorer nos résultats. Une première consiste d'améliorer notre technique de partitionnement qui présente encore quelques inconvénients, notamment l'obligation de faire un compromis entre fiabilité, réactivité et connectivité. Une deuxième piste consiste à réduire la propagation des erreurs liées aux positions estimées en utilisant par exemple les filtres de Kalman, ou encore la méthode des moindres carrés pour améliorer l'estimation de la position en prenant en compte la position précédente mais aussi regarder l'impact de la mobilité des nœuds sur les performances de *L-Libre*. La troisième voie consiste à une réadaptation de notre approche collaborative de détection et de poursuite dans le cas de cibles multiples. En définitive, le suivi d'objets multiples revient à résoudre conjointement deux problèmes : l'association des données et l'estimation par un algorithme de filtrage particulaire. Et enfin proposer une stratégie d'économie d'énergie qui vise à allonger la durée de vie globale des réseaux mobiles.

Publications

Revues internationales avec comité de lecture

J. F. Myoupo, A. Ould Cheikhna and I. Sow A randomized clustering of anonymous wireless ad hoc networks with an application to the initialization problem, *To appear in Journal of Supercomputing, 2009*

J. F. Myoupo and I. Sow Clustering in mobile ad hoc networks with some nodes having the same weight, *Studia Informatica Universalis Vol 6.1 pages 1-13. ISBN 2-7056-6734-2, 2008 Edition Hermann.*

Conférences internationales avec comité de sélection

J. F. Myoupo and I. Sow Data gathering in a virtual architecture for wireless sensor networks with some empty clusters, *In Proceedings of International Conference Wireless Networks (ICWN) Volume I pages 75-81, Las Vegas, Nevada USA, July 2008. .*

M. Bui, J.F. Myoupo and I. Sow, An operational model for model target tracking in wireless sensor networks. *In Proceedings of* 2^{nd} *IEEE International Symposium on Wireless Pervasive Computing, Puerto Rico Feb 2007.*

J. F. Myoupo and I. Sow Self-localization and connectivity maintenance scheme for target tracking in wireless sensor networks, *In Proceedings of the Seventh IASTED International Conferences Wireless and Optical Communications,Montreal, Quebec Canada, June 2007.*

R. Mellier, J.F. Myoupo and I. Sow, GPS-Free geocasting algorithms in mobile ad hoc networks, *In IEEE International Conference on Wireless and Mobile Communications (ICWMC),pp.38, Romania July 2006.*

Bibliographie

[1] A. Caruso, S. Chessa, S. De and A. Urpi. Gps-free coordinate assignement and routing in wireless sensor networks,. Miami FL USA, June 2005.

[2] B.M. Blum, T. He, C. Huang and J.A. Stankovic. Range-free localization and its impact on large scale sensor networks. In *Trans. on Embedded Computing Sys*, number 4, 2005.

[3] B.R. Badrinath and D. Niculescu. Ad hoc positioning system. In *Proceedings of GLOBECOM*, San Antonio, Novembre 2001.

[4] S. Basagni. Distributed clustering for ad hoc networks. In *Proccedings of International Symposium on Parallel Architectures, Algorithms, and Networks*, pages 310–315, SPAIN, 1999. IEEE Computer Society,.

[5] P. Bhagwat and C. Perkins. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers,. In *Association for Computing Ma-chinery's Special Interest Group on Data Communication*, pages 234–244, 1994.

[6] J. Caffery and D. Goyal. Partitioning avoidance in mobile ad hoc networks using network survivability concepts,. In *In Proc IEEE Int. Symp. Computers and Communications ISCC, Taormina*, pages 553–558, 2002.

[7] C.C. Han, A. Savvides and M.B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors,. In *In Proceedings of the 7th annual international conderence on Mobile computing and networking*, pages 166–170, 2001.

[8] T. Clausen and P. Jacquet. Optimized link state routing protocol,. In *In IETF Internet Draft, draft-ietf-manet-olsr-11.txt*, July 2003.

[9] M. Corson and V. Park. A highly adaptive distributed routing algorithm for mobile wireless networks,. In *In Proceedings of IEEE INFOCOM*, April 1997.

[10] M. Corson and V. Park. A performance comparison of the temp orally-ordered routing algorithm and ideal link state routing,. In *In Proceedings of of ISCC 1998*, July 1998.

[11] D.J. Baker, A. Ephremides and J.A. Flynn . The design and simulation of a mobile radio network with distributed control. In *IEEE Journal on Selected Areas in Communication SAC-2, 1*, pages 226–237, Janvier 1984.

[12] D.J. Baker, A. Ephremides and J.E. Wieselthier . A design concept for reliable mobile radio networks with frequency hopping signaling. In *Proceedings of the IEEE 75,1*, pages 56–737, Janvier 1987.

[13] A. Ephremides. Design concepts for mobile-user radio network. In *Computers Electrical Engineering 10,2*, pages 49–64, 1983.

[14] D.J. Baker and A. Ephremides. The architectural organization of a mobile radio network via a distributed control. In *IEEE Transactions on Communication COM-29, 11*, pages 1694–1701, Novembre 1981.

[15] F. Benbadis, M. Dias de Amorim and S. Fdida. Quelques applications des réseaux de capteurs. In *In Journées Scientifiques Techniques CetMet*, volume 38, pages 393–422, 2004.

[16] F. Bruyaux, M. Duque-Anton and P. Semal. Measuring the survivability of network : connectivity and resconnectivity,. In *European Transaction of Telecommunications*, volume 11, pages 149–159, 2000.

[17] W. Feller. An introduction to probability theory and its applications. *3rd edition John WILEY and Sons*, 1968.

[18] G.G. Finn. Routing and addressing problem in large metropolitain-scale internetworks,. In *IST res. rep ISU/RR*, pages 87–180, 1987.

[19] H. Balakrishnan, A.K. Miu, N.B. Priyantha and S. Teller. The criket compass for context-aware mobile applications,. In *In Proceedings of 7nd annual international Conference on Mobile computing and networking*, pages 1–14, New York ,USA, 2001. ACM Press,.

[20] H. Park, A. Savvides and B. Srivastava. The bits and flops of n-hop multilateration primitive for node localization problems. In *In Proceedings of the 1st ACM International workshop on Wireless sensor networks and applications*, pages 112–121, New york ,USA, 2002. ACM press,.

[21] L. Iannone and S. Fdida. *Meshdv : A distance vector mobility-tolerant routing protocol for wireless mesh networks.* Santorini Greece, July 2005.

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

[23] T. Imelinski and J. Navas. Geocast-geographic addressing and routing,. In *In Proceedings of ACM/IEEE MOBICOM'97*, volume 3, pages 66–76, 1997.

[24] J. Leonard, D. Moore, D. Rus and S. Teller. Robust distributed network lcalization with noisy range measurements,. In *In Proceedings of 2nd International Conference on Enbedded networked sensor system*, pages 50–61, New York ,USA, 2004. ACM Press,.

[25] D.B. Johnson and D.A. Maltz. The dynamic source routing protocol for multihop wireless ad hoc networks. In *Addison-Wesley*, pages 139–172, 2001.

[26] K. Chen, K. Nahrsted and T.S. Shah. Cross-layer design for data accessibily in mobile ad hoc networks,. In *In Wireless Personnal Communications*, volume 21, pages 49–76, Netherlands, 2002. KluwerAcademic Publishers,.

[27] K. Alagha, H. Badis, A. Munaretto and G. Pujolle. A link-state qos routing protocol for ad hoc networks. In *In Proceedings of the 4th IFIP IEEE International Conference on Mobile and Wireless Communication Networks*, Sweden, Sept 2002.

[28] W.J. Kaiser and G.J. Pottie. Wireless integrated sensor networks. In *Commnications of the ACM*, volume 43, pages 51–58, 2000.

[29] B. Karp and H.T. Kung. Gpsr :greedy perimeter stateless routing for wireless networks,. In *In Proceedings of ACM/IEEE MOBICOM'00*, August 1997.

[30] L. Kleinrock and H. Takagi. Optimal transmission ranges for randomly distributed packet radio terminals,. In *IEEE Transactions on Communications*, volume 32, pages 246–257, 1984.

[31] Y.B. Ko and N.H. Vaidya. Location-aided routing in mobile ad hoc networks,. In *In Proceedings of ACM/IEEE MOBICOM'98*, pages 66–75, August 1998.

[32] L. Doherty, L.E. Ghaoui and H.S.J. Pister. Convex position estimation in wireless sensor networks. In *In Proc. INFOCOM*, Anchorage AK, April 2001.

[33] B. Li and K.H. Wang. Efficient and guaranteed service coverage in partitionable mobile ad hoc networks,. In *INFOCOM*, 2002.

[34] B. Li and K.H. Wang. Group mobility and partition prediction in wireless ad hoc networks,. In *IEEE ICC*, April 2002.

[35] X. Lin and I. Stojmenovic. Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks,. In *IEEE Transactions on Parallel and Distributed Systems*, volume 12, pages 1023–1032, October 2001.

[36] M. Gerla and C.R. Lin. Adaptive clustering for mobile wireless networks. *Journal on Selected Ares in Communication*, 15(7) :1265–1275, September 1997.

[37] M. Gerla and J.T.C. Tsaï . Multicluster, mobile multimedia radio network. *Wireless Networks*, 1(3) :255–265, 1995.

[38] M. Hauspie, M. Jorgic, D. Simplot-Ryl and I. Stojmenovic. Localized algorithms for detection of critical nodes and links for connectivity in ad hoc networks. In *In Proc the 3th Annual IFIP Mediterranean Ad hoc Networking Workshop MED'HOC NET*, Turkey, 2004.

[39] M. Lewis, R. Ogier and F. Templin. *Top ology Dissemination Based on Reverse-Path Forwarding (TBRPF),*. IETF RFC-3684, February 2004.

[40] N. Bulusu, D. Estrin and J. Heidemann. Gps-less low cost outdoor localization for very small devices,. In *IEEE Personal Communications*, volume 7, pages 28–34, Oct 2000.

[41] N. Bulusu, D. Estrin and J. Heidemann. Gps-less low cost outdoor localization for very small devices. In *IEEE Personal Communications*, volume 7, pages 28–34, 2002.

[42] K. Nakano and S. Olariu. Randomized initialization protocols for radio networks,. In *In Handbook of Wireless Networks and Mobile Computing*, pages 195–218, New York, 2002.

[43] K. Nakano and S. Olariu. Uniform leader election for radio networks,. In *In IEEE Transactions on Parallel and Distributed Systems*, number 13, pages 516–526, 2002.

[44] B. Nath and D. Niculescu. Ad hoc positioning System Using AoA,. In *In Proceedings of IEEE Infocom*, 2003.

[45] P. Bah and V.N. Padmanabhan. Radar : An in building rf-based user location and tracking system. In *Proceedings of GLOBECOM*, pages 775–784, 2000.

[46] R. Axel, P. Enge and B.W. Parkinson. Global positioning system : Theory and application. In *American Institute of Aeronautics and Astronautics,*, 1996.

[47] R. Axel, S. Basagnie, M. Conti, P. Enge, S. Giordano and W. Parkinson. Mobile ad hoc networking. In *Wiley-IEEE Press*, August 2004.

[48] R. Jain, A. Puri and R. Sengupta. Geographical routing using partial information for wirelessad hoc networks,.

[49] J. Rabaey and C. Saverese. Robust positining algorithms for distributed ad-hoc sensor networks,. In *In USENIX technical annual conference*, pages 317–328, Monterey CA, June 2002.

[50] V. Ravelomanana. Optimal initialization and gossiping algorithms for random radio networks. *IEEE TPDS*, 18 :17–28, 2007.

[51] S. Basagni and R. Ghosh . Limiting the impact of mobility on ad hoc clustering. In *Proccedings of the Second ACM International Workshop on Performance Evaluation of Wireless Ad hoc, Sensor & Ubiquitous Networks*, pages 197–204. IEEE Computer Society,, Octobre 2005.

[52] S. Capkun, M. Hamdi and J. Hubaux. Gps-free positionning in mobile ad-hoc networks,. In *In Proceedings of 34th Anuual hawaii International Conference on System Siences (HICSS-34)*, volume 9, page 9008, Washington DC ,USA, 2001. IEEE Computer Society,.

[53] S.R. Das, C. Perkins and E.M. Royer. Ad hoc on-demand distance vector routing,. In *In IETF Internet Draft, draft-ietf-manet-AODV-13.txt*, February 2003.

[54] R. Tarjan. Depth first search and linear graph algorithms. In *SIAM J. Computing*, volume 1, pages 146–160, 1972.

[55] V. Bychkovskiy, J. Elson, D. Estrin and L. Girod. Locating tiny sensors in time and space : A case study. In *In Proc. International Conference on Computer Design (ICCD)*, Freiburg Germany, 2002.

[56] H. Frey and I. Stojmenovic. On delivery guarantees of face and combined greedy-face routing algorithms in ad hoc and sensor networks,. In *The Twelfth ACM Annual International Conference on Mobile Computing and Networking MOBICOM*, pages 390–401, Los Angeles, Sept 2006.

Listes des abbréviations

ADSL	Asymmetric Digital Subscriber Line
AODV	Ad-hoc On Demand Distance Vector
AOA	Angle Of Arrival
APS	Ad hoc Positioning System
DCA	Distributed Clustering Algorithm
DMAC	Distributed and Mobility-Adaptive Clustering
EEPROM	Electrically-Erasable Programmable Read-Only Memory
ETSI	European Telecommunications Standards Institute
GPS	Global Positioning System
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
MANET	Mobile Ad-hoc NETwork
MPR	MultiPoint Relay
OLSR	Optimized Link State Routing
TDOA	Time Difference Of Arrival
QoS	Quality of Service
RSSI	Receive Signal Strength Indicator
UWB	Ultra-Wide Band
WAN	Wide Area Network
WAP	Wireless Application Protocol
Wi-Fi	Wireless Fidelity
WEP	Wireless Encription Protocol
WLAN	Wireless Local Area Network
WMAN	Wireless Metropolitan Area Network
WRAN	Wireless Regional Area Network
WWAN	Wireless Wide Area Network

...

Table des figures

Liste des tableaux