UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

LA DIFFUSION DE MESSAGES DANS LES RÉSEAUX RADIO ANONYMES

MÉMOIRE
PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE

PAR

KATIA LARRIVÉE

OCTOBRE 2006

UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

Département d'informatique et d'ingénierie

Ce mémoire intitulé :

LA DIFFUSION DE MESSAGES DANS LES RÉSEAUX RADIO ANONYMES

présenté par

Katia Larrivée

pour l'obtention du grade de maître ès science (M.Sc.)

a été évalué par un jury composé des personnes suivantes :

Dr. Andrzej Pelc ..... Directeur de recherche

Dr. Luigi Logrippo ..... President du jury

Dr. Rokia Missaoui ..... Membre du jury

Mémoire accepté le : 20 octobre 2006

À mon futur mari Etienne.

À ma soeur et à mes parents.

Remerciements

Je tiens à remercier Andrzej Pelc pour la précieuse collaboration offerte tout au long de mes études de maîtrise. Merci pour les conseils, les encouragements et le soutien tout au long de ce mémoire. J'aimerais tout particulièrement aussi remercier mon conjoint Etienne pour le soutien. Merci aussi à ma famille et mes amis pour leurs encouragements.

Merci de tout mon coeur

Table des matières

Remerciements i

Liste des figures iv

Liste des tableaux vi

Résumé vii

1 Identification du problème et motivation 1

1.1 Problème 1

1.2 Motivation et importance de travailler sur ce problème 2

2 Revue de la littérature 6

2.1 Les modèles de réseaux radio 6

2.1.1 Algorithmes déterministes 7

2.1.2 Algorithmes aléatoires 12

2.2 La technologie des réseaux sans fil 13

2.2.1 Les catégories de réseaux sans fil 13

2.2.2 Les réseaux ad hoc 15

2.3 Comparaison des résultats connus avec le problème du mémoire 15

3 Présentation des résultats de la recherche 17

3.1 Introduction 17

3.2 L'impossibilité d'atteindre des noeuds spéciaux dans un réseau radio anonyme 18

3.3 L'algorithme informant les noeuds accessibles 23

4 Algorithme de diffusion de messages dans les réseaux radio anonymes de type grille rectangulaire **26**
4.1 Introduction 26
4.2 Les axes et les demi-axes 29
4.3 Les différents cas possibles 30
4.4 La notation et terminologie utilisées 32
4.5 Description de l'algorithme 36
4.5.1 La procédure élection des axes internes 36
4.5.2 La procédure élection des axes externes 38
4.5.3 La procédure retour du message à la source 39
4.5.4 La procédure élection du plus grand ou plus petit chef champion . . . 41
4.5.5 La procédure synchronisation des chefs de lignes 42
4.5.6 La procédure diffusion par balayage dans l'axe interne 44
4.5.7 La procédure diffusion du message par balayage 45
4.5.8 La procédure élection des chefs-coins spéciaux 47
4.5.9 La procédure multiplexage 49
4.5.10 Concaténation des procédures 51
4.5.11 L'algorithme maître : diffusion dans la grille rectangulaire 54
4.6 Les résultats 55
4.7 Conclusion 56

5 Simulations de l'algorithme de diffusion **57**
5.1 Introduction 57
5.2 Description de l'application 58
5.3 Les simulations 59
5.4 Conclusion 62

6 Conclusion **63**

A Le code source partiel de l'application (grille.java) **66**

Bibliographie **94**

Liste des figures

1.1 Problème du terminal caché 3
1.2 Problème du terminal exposé 4

2.1 Transmission par vagues 9

3.1 Graphe avec un automorphisme autre que l'identité 18
3.2 Exemples de noeuds semblables a et a' 21
3.3 Exemple d'un ensemble spécial 22
3.4 Ensemble spécial dans une grille carrée impaire avec la source centrée 23
3.5 Ensemble spécial dans une grille carrée impaire avec la source située sur une diagonale 24
3.6 Ensemble spécial dans une grille carrée avec la source située dans un coin . . 25

4.1 Les axes internes d'une grille rectangulaire 29
4.2 Les axes externes d'une grille rectangulaire 29
4.3 Les demi-axes d'une grille rectangulaire 30
4.4 Deux axes de longueur différente 30
4.5 Deux axes de même longueur 31
4.6 Deux demi-axes de longueur différente 31
4.7 Deux demi-axes de même longueur 32
4.8 Quatre demi-axes de même longueur 32
4.9 Les demi-axes égaux sont adjacents - égaux en coin 33
4.10 Les demi-axes égaux sont opposés - égaux en face à face 33
4.11 Au moins un demi-axe de longueur unique 34
4.12 La procédure élection des axes internes 37
4.13 La procédure élection des axes externes 39

4.14 La procédure élection des axes externes 39
4.15 La procédure retour du message à la source 40
4.16 Élection du plus grand ou plus petit chef champion 42
4.17 Élection du plus grand ou plus petit chef champion 43
4.18 La procédure synchronisation des chefs de ligne 43
4.19 La procédure diffusion dans l'axe interne 44
4.20 La procédure diffusion du message par balayage 45
4.21 Grille carrée impaire avec la source située sur une diagonale 46
4.22 Élection dans une grille carrée impaire avec la source située sur une diagonale 47
4.23 Élection dans une grille carrée impaire avec la source située sur une diagonale 48
4.24 Élection dans une grille carrée impaire avec la source située sur une diagonale 49
4.25 Synchronisation dans une grille carrée impaire avec la source située sur une diagonale 50
4.26 Balayage dans une grille carrée impaire avec la source située sur une diagonale 51

5.1 Relation entre le rayon des grilles et le nombre de rondes divisé par le rayon de la grille 61

Liste des tableaux

5.1 Résultats des simulations lorsque la position de la source est connue 60
5.2 Résultats des simulations lorsque la position de la source est inconnu 60
5.3 Nombre de noeuds non informés 62

Résumé

Nous considérons des réseaux de communication de type radio, modélisés par des graphes non-orientés.

Un noeud envoie un message à tous ses voisins. Lorsqu'un noeud reçoit plus d'un message en même temps, il n'entend rien. Les noeuds n'ont pas d'étiquettes distinctes, l'action d'un noeud dépend donc uniquement des messages reçus antérieurement. Une source doit transmettre un message à tous les noeuds accessibles du réseau, c'est-à-dire à tous les noeuds qui peuvent entendre le message.

Nous nous intéressons à savoir dans quels cas nous pouvons obtenir une diffusion complète (tous les noeuds reçoivent le message) ? Quels noeuds ne pourront jamais être atteints dans un réseau donné ? Quels sont les algorithmes de diffusion possibles et quelle est leur efficacité ? Nous étudierons ces questions principalement pour une classe de topologies de réseaux spécifiques : les grilles rectangulaires.

Nous proposerons un algorithme qui permettra d'informer tous les noeuds accessibles. Nous présenterons aussi une application implantant cet algorithme et réaliserons des simulations.

Ce projet vise principalement à étudier la faisabilité de la diffusion. Dans le cas où une diffusion complète serait impossible, il faudra déterminer l'ensemble de noeuds du réseau qui ne peuvent pas être informés et informer tous les autres noeuds.

Abstract

We consider radio communication networks modeled as non-oriented graphs. A node sends a message to all its neighbors. When a node receives more than one message in the same time, it doesn't hear anything. Nodes of the network do not know its topology and they do not have distinct labels. The decision made by a node to transmit or not can only be based on the messages previously heard. The source starts the broadcasting by sending a message. We want all accessible nodes to receive the message. A node is accessible if it can be reached during the broadcasting.

We want to know in which cases we can have a complete broadcast (all nodes are reached) ? Which nodes can never be reached in a certain network ? We will study theses questions for a specific network class of topologies : the grids.

We'll propose an algorithm that can reach all accessible nodes. We also present an application implementing this algorithm and perform simulations.

Chapitre 1

Identification du problème et motivation

1.1 Problème

Un réseau radio (sans fil ou "wireless network" en anglais) est un réseau dans lequel au moins deux terminaux peuvent communiquer sans liaison filaire. Grâce aux réseaux sans fil, un utilisateur a la possibilité de rester connecté tout en se déplaçant dans un périmètre géographique plus ou moins étendu. C'est la raison pour laquelle on peut parler de la mobilité de ces réseaux.

Les réseaux sans fil sont basés sur une liaison utilisant des ondes radio-électriques (radio et infrarouges). Il existe plusieurs technologies se distinguant par la fréquence d'émission utilisée ainsi que par le débit et la portée des transmissions.

Les réseaux sans fil permettent de relier très facilement des équipements distants d'une dizaine de mètres à quelques kilomètres. De plus, l'installation de tels réseaux ne demande pas de lourds aménagements des infrastructures existantes comme c'est le cas avec les réseaux filaires (équipements des bâtiments en câblage et connecteurs), ce qui a valu un développement rapide de ce type de technologies.

Dans ce mémoire, nous avons considéré les réseaux de communication de type radio. Ce réseaux sont modélisés par des graphes non-orientés. Les différents émetteurs / récepteurs sont modélisés par des noeuds et leur possibilité de communiquer ensemble est modélisée par des arêtes.

Lorsqu'un noeud envoie un message, il l'envoie à tous ses voisins dans le graphe au même moment. Dans ce cas, il ne peut entendre aucun message. Lorsqu'un noeud reçoit plus d'un message en même temps, il n'entend rien. Ceci est causé par l'interférence entre les messages.

Une source doit transmettre un message à tous les noeuds accessibles du réseau. Un noeud est accessible s'il peut être informé par un algorithme déterministe (possiblement dépendant du réseau). Au début, seule la source connaît le message. Comme les noeuds n'ont pas d'étiquettes distinctes, l'action d'un noeud dépend uniquement des messages reçus antérieurement.

Nous nous intéressons à savoir dans quels cas nous pouvons obtenir une diffusion complète (tous les noeuds reçoivent le message) ? Quels noeuds ne pourront jamais être atteints dans un réseau donné ? Quels sont les algorithmes de diffusion possibles et quelle est leur efficacité ? Nous étudierons ces questions pour une topologie de réseaux spécifiques : les grilles. La source peut être située dans un noeud arbitraire de la grille rectangulaire et sa position peut être connue ou inconnue.

Le projet vise principalement la question de faisabilité de la diffusion et de la détermination de l'ensemble de noeuds d'un réseau qui ne peuvent être atteints par aucun algorithme déterministe de diffusion.

1.2 Motivation et importance de travailler sur ce problème

L'intérêt concernant la communication sans fil s'est considérablement développé avec l'émergence des réseaux hertziens (un réseau hertzien est un réseau qui utilise les ondes électromagnétiques comme médium de communication). Nous utilisons dans notre vie de tous les jours les téléphones cellulaires, les technologies sans fil dans le domaine informatique (la technologie Bluetooth, la norme 802.11) ou la mise en place de satellites. La demande des réseaux sans fil augmente de jour en jour en apportant avec elle de nouveaux problèmes liés aux modèles de communication.

En contrepartie se pose le problème de la réglementation relative aux transmissions radio-électriques. En effet, les transmissions radio-électriques servent pour un grand nombre d'applications (militaires, scientifiques, amateurs). Malheureusement, elles sont très sensibles aux interférences. C'est la raison pour laquelle une réglementation est nécessaire dans chaque pays afin de définir les plages de fréquence et les puissances auxquelles il est possible d'émettre pour chaque catégorie d'utilisation.

De plus, les ondes hertziennes sont difficiles à confiner dans une surface géographique restreinte, il est donc facile pour un pirate d'écouter le réseau si les informations circulent en clair (c'est le cas par défaut). À l'instar d'Ethernet, il existe des méthodes d'accès aléatoire à une interface radio.

Il y a deux autres problèmes qui concernent directement notre projet de recherche et touchent à la notion de collisions dans les réseaux sans fil : Le problème du terminal caché et le problème du terminal exposé.

Problème du terminal caché : La situation correspondant à ce problème est illustrée à la figure 1.1. Ici, le processeur A transmet un message au processeur B. Le processeur C veut également transmettre un message à B. Il écoute le canal pour savoir si celui-ci est libre. Comme la portée de communication de A ne lui permet pas d'atteindre C, ce dernier va penser que le canal est libre, transmettre son message, et venir brouiller le message que A est en train d'envoyer à B. On voit donc que C n'a aucun moyen de savoir si A est en train de transmettre car il est caché. S'il transmet, il y aura une collision de messages sur B.

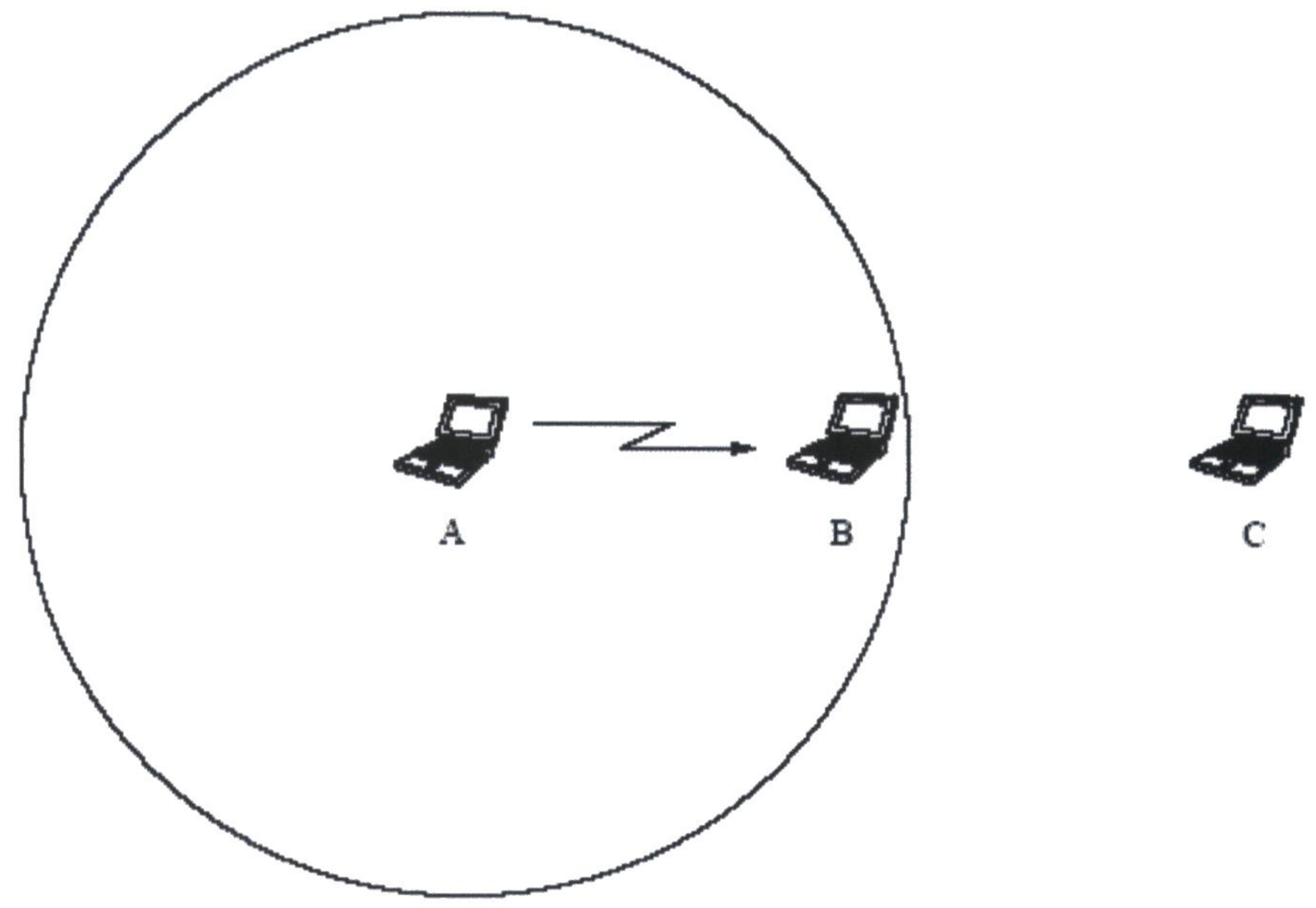

FIG. 1.1 – Problème du terminal caché

Problème du terminal exposé : Ce problème, illustré à la figure 1.2 est l'inverse du précédent. Ici, le processeur A transmet un message à D. Pendant ce temps, B désire trans-

mettre un message à C et écoute le canal. Il entend A (car il est à portée de communication) et ne transmet donc pas son message à C. Or, cette transmission ne brouillerait pas le message de A à D car il n'est pas à portée de B. Le débit utilisé est donc diminué car B ne transmet pas alors qu'il aurait pu le faire.

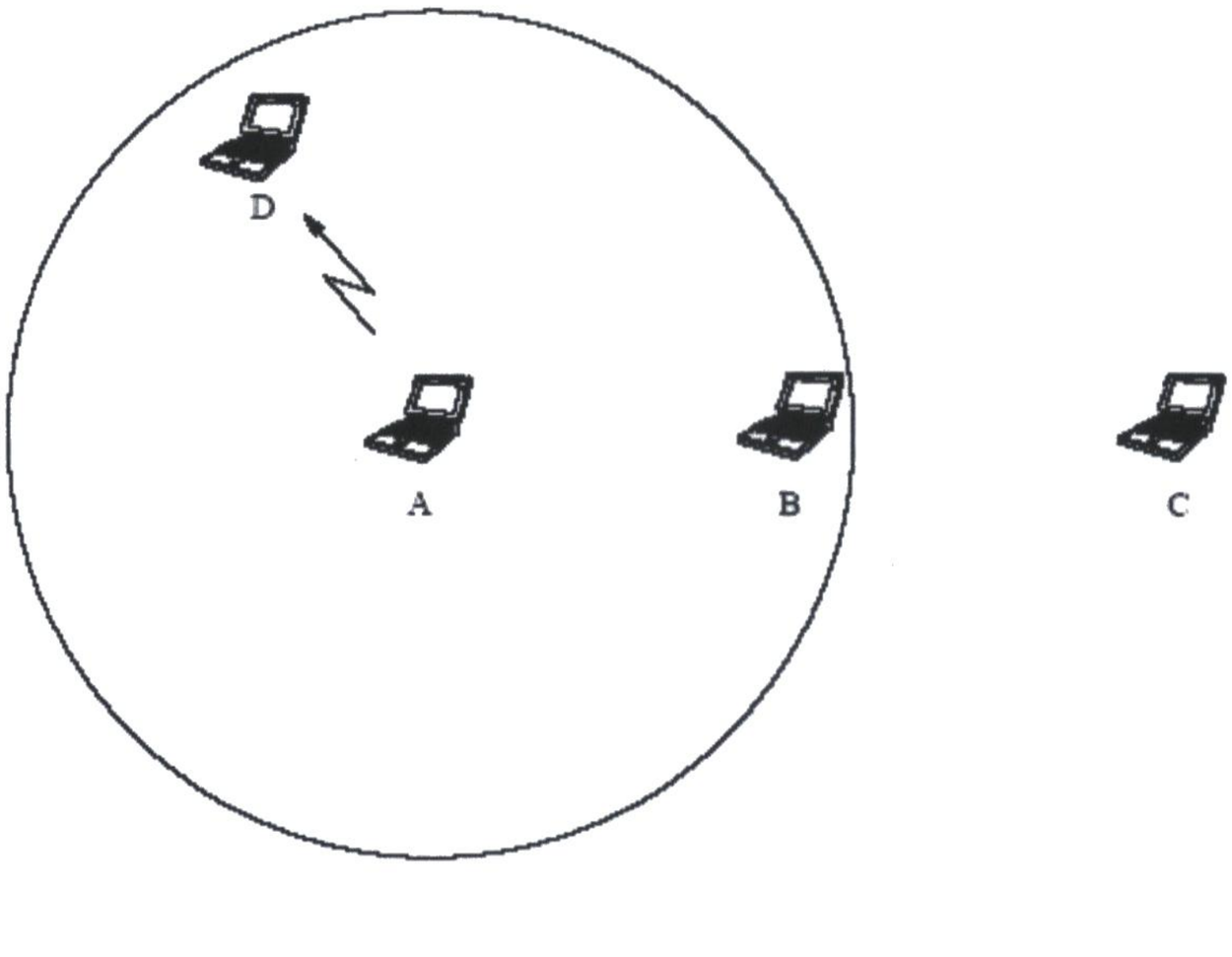

FIG. 1.2 – Problème du terminal exposé

Ces deux problèmes illustrent le rôle des collisions dans la communication sans fil. Les collisions augmentent considérablement la difficulté de construction des algorithmes de communication dans les réseaux radio. Trop de noeuds transmettant à la fois augmentent le nombre de collisions et cela réduit la vitesse de communication. Alors que trop peu de transmissions diminuent le débit par rapport au débit optimal.

Un autre problème étroitement lié à notre recherche est celui de l'anonymat des noeuds. Les noeuds peuvent être anonymes pour plusieurs raisons :

1. La complexité d'administrer toutes les adresses (ou l'autoconfiguration du réseau)
2. Le coût
3. La nécessité d'être anonyme (opération militaire)
4. La production en masse (quantité énorme de noeuds identiques).

L'exemple parfait de noeuds qui pourraient être anonymes sont les sondes. Il est beaucoup plus intéressant de déployer des milliers de minuscules sondes qui communiquent ensemble sans étiquette. Les coûts reliés à l'implantation d'identificateurs uniques pour chacune des sondes peuvent être énormes.

Par contre, l'utilisation de noeuds anonymes dans les réseaux radio ajoute certaines difficultés :

1. Il est impossible de demander à un noeud spécifique (en se servant de son étiquette) de transmettre.
2. Afin de décider si un noeud doit transmettre ou non, seuls son histoire (les messages qu'il a reçus précédemment) et possiblement son degré pourront être utilisés.
3. Certains noeuds seront inaccessibles c'est-à-dire qu'aucun algorithme déterministe (même connaissant le réseau) ne peut les informer.

Chapitre 2

Revue de la littérature

2.1 Les modèles de réseaux radio

Dans la majorité des travaux concernant l'algorithmique des réseaux radio, le réseau est modélisé par un graphe G = (V, E) avec V l'ensemble des noeuds du graphe (à chaque station du réseau correspond un noeud dans le graphe) et E l'ensemble des arcs donnant les possibilités de communication directe entre les stations (dans certains cas les communications sont bidirectionnelles et dans d'autres cas elles sont unidirectionnelles). L'ensemble $N(u) = \{v \in V : (v, u) \in E\}$ est appelé le voisinage de u et tous les noeuds $v \in N(u)$ sont appelés les voisins de u. La distance entre deux sommets est la plus courte longueur qui les relient. Le diamètre D d'un graphe G est la plus grande distance entre deux sommets du graphe.

Un des modèles principaux est celui où le graphe représentant le réseau est obtenu à partir des distances physiques entre les stations. Si on pose d(uv) comme étant la distance physique séparant les noeuds u et v, et R le rayon de communication des noeuds, l'ensemble E se définit par : $E = \{(u, v)\} \in V^2 | d(uv) < R\}$.

Cependant, surtout dans le contexte urbain, des obstacles peuvent empêcher la communication dans certaines directions, ce qui implique la nécessité de modéliser le réseau par des graphes plus généraux.

La communication dans les réseaux radio modélisés par des graphes est basée sur les deux principes suivants :

1. Lorsqu'un noeud envoie un message, celui-ci est envoyé à tous ses voisins

2. Un noeud u reçoit un message de son voisin v dans une ronde t si et seulement si v est l'unique noeud dans N(u) qui transmet dans la ronde t

Une des tâches principales, dans les réseaux radio, est la diffusion. Un noeud doit envoyer un message à tous les autres noeuds du réseau. Considérons un graphe G avec un diamètre D et une source S. Nous voulons compléter une diffusion dans G. Peu importe le réseau, peu importe l'algorithme, une première borne inférieure s'impose sur le temps de la diffusion : $\Omega(D)$ où D est le diamètre du graphe. En effet, peu importe l'algorithme, il faudra au minimum D rondes pour transmettre le message à tous les noeuds. Cette borne n'est, en général, pas possible à atteindre. Pour y arriver, il faudrait qu'à chaque ronde le message progresse toujours d'un pas vers le noeud le plus éloigné, ce qui est souvent impossible à cause des collisions. En 1991, les auteurs de [1] ont étudié la question. Ils ont augmenté la borne inférieure à $\Omega(D + \log^2 n)$ pour une certaine famille de graphes de diamètre D et de n noeuds avec n'importe quel algorithme (déterministe ou aléatoire).

Les algorithmes de diffusion dans un réseau radio peuvent être de deux natures, déterministes ou probabilistes. Dans ce qui suit, nous allons faire un bref survol des résultats concernant ces deux types d'algorithmes

2.1.1 Algorithmes déterministes

Diffusion dans les réseaux centralisés

Modèle Le premier cas traité est celui du réseau centralisé, c'est-à-dire, un réseau avec un moniteur central qui contrôle ce qui se passe dans le réseau. C'est comme s'il y avait un arbitre qui envoie les commandes aux différentes stations. Si les noeuds du réseau ont une connaissance complète de sa topologie, ils peuvent simuler les actions du moniteur central de façon distribuée sans que l'existence d'un tel moniteur réel soit nécessaire.

Chacun des noeuds possède une étiquette unique et chacun des noeuds connaît sa propre identité. Lorsque nous parlons de la diffusion centralisée, on suppose que la topologie complète du graphe est connue.

Résultats En 1985, les auteurs de [4] ont écrit un article sur l'analyse des problèmes et sur les protocoles de la diffusion des réseaux radio. Les auteurs proposaient d'utiliser un graphe (orienté ou non) pour la modélisation des réseaux radio. Ils ont expliqué qu'une solution

simple comme l'inondation ne pourrait pas compléter une diffusion dans un graphe quelconque. Ils ont aussi énuméré les conditions nécessaires d'un protocole optimal. Un protocole optimal doit minimiser au moins une de ces mesures :

1. D maximum : le maximum de temps nécessaire pour effectuer une diffusion complète.
2. D moyen : le temps moyen nécessaire pour que le message soit connu par chaque noeud.

Ils ont prouvé que la minimisation de chacun de ces deux critères étaient un problème NP-complet.

Le premier grand résultat obtenu fut un algorithme de diffusion qui fonctionne en temps $O(D \ \log^2 n)$. L'idée principale de cet algorithme [5] est de faire la diffusion par vagues. Le message progresse dans le réseau par vagues de transmissions. Afin de savoir qui va transmettre, à chaque ronde, il y a élection des porte-parole. La procédure pour vérifier à chaque ronde tous les cas possibles (2^n cas à vérifier) utilise un algorithme d'élection dans les sous-ensembles. Les porte-parole choisis sont ceux qui en transmettant, parviennent à maximiser le nombre de bonnes transmissions durant une ronde. (Ce résultat a été publié en 1991) Voici les trois critères que les auteurs voulaient satisfaire avec leur algorithme :

1. Minimiser les délais de transmission
2. Utiliser beaucoup de parallélisme
3. Localiser l'exécution de la transmission.

Voici, la figure 2.1 d'un réseau qui utilise la procédure de l'élection du porte-parole :

Durant la première ronde, le noeud 1 informe ses voisins (2, 3, 4). Ils sont maintenant des transmetteurs potentiels et les noeuds 5, 6, 7 et 8 sont des receveurs potentiels. Pour maximiser les bonnes réceptions, seuls les noeuds 2 et 4 devront transmettre pendant la ronde 2.

Suite à cet article, les auteurs de [13] ont prouvé que la diffusion aléatoire peut se faire en temps $O(D + \log^5(n))$, ce qui implique une diffusion optimale lorsque $D = \Omega(\log^5 n)$. En utilisant le schéma de [5], nous obtenons une diffusion déterministe qui travaille en temps $O(D + \log^6(n))$. L'idée de l'algorithme est la suivante : le réseau est divisé en groupes (un groupe est constitué de plusieurs niveaux de fouille en largeur consécutifs). Dans chaque groupe, il y a des secteurs de sorte que :

1. Chaque secteur a un petit diamètre

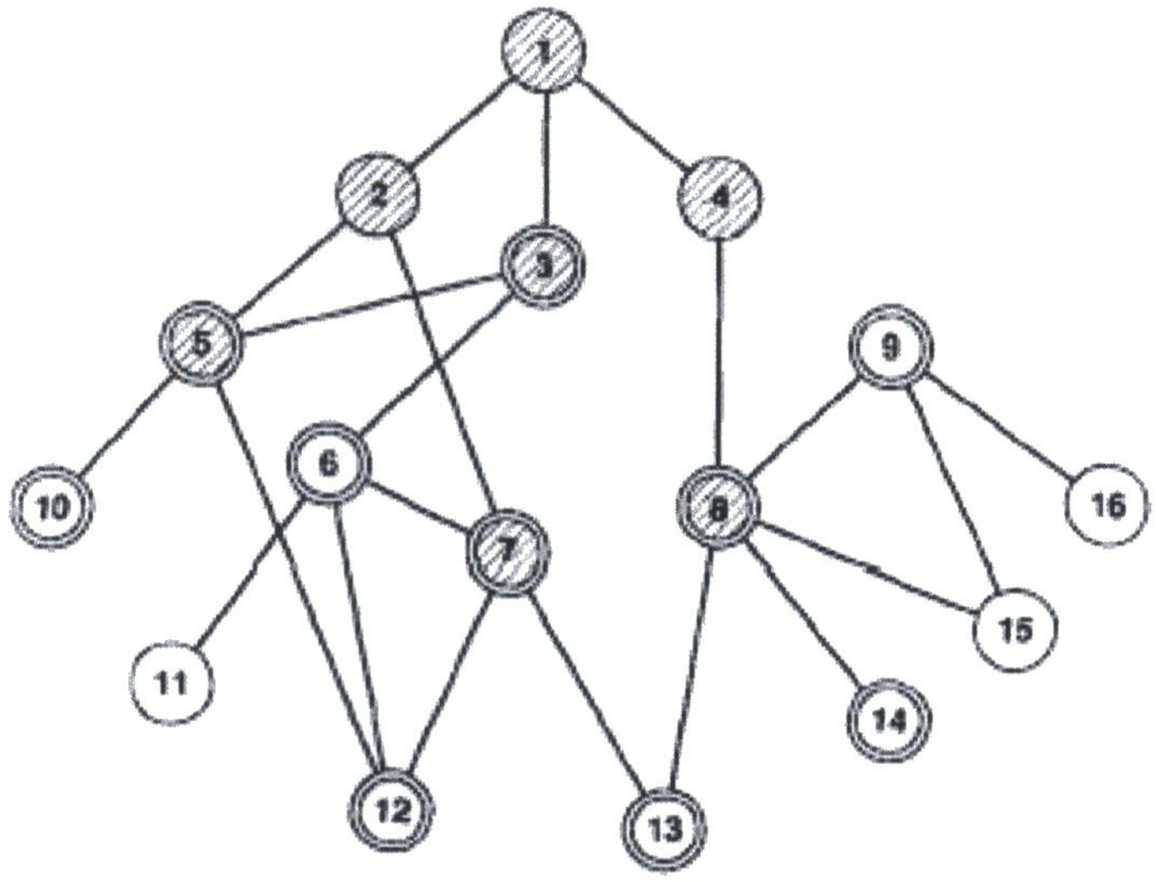

FIG. 2.1 – Transmission par vagues

2. L'union de tous les secteurs couvre le groupe
3. Le graphe des secteurs peut être coloré en $O(\log n)$ couleurs.

L'objectif est d'informer un noeud de chaque secteur qui doit, à son tour, informer les autres noeuds de son secteur. Les secteurs doivent avoir le diamètre le plus petit que possible afin que le message se transmette rapidement dans les secteurs. En même temps, lorsqu'un chef de secteur fait la diffusion dans son secteur, tous les autres secteurs ayant la même couleur peuvent le faire sans risque de collisions. Cette technique permet beaucoup de parallélisme dans la diffusion.

La prochaine étape à franchir fut de trouver un algorithme déterministe travaillant en temps polynomial qui permet de construire un schéma de diffusion rapide pour tous les réseaux radio. Le résultat de [15] est un schéma de diffusion travaillant en temps $O(D \log n + \log^2(n))$ pour n'importe quel graphe de n noeuds avec un diamètre de D. En combinant ce schéma avec la méthode de Elkin et Kortsarz ainsi que Gaber et Mansour, nous obtenons un algorithme de diffusion déterministe qui travaille en $O(D + \log^4(n))$.

À ce jour, le dernier résultat publié est celui de [14] qui travaille en temps : $O(D+\log^3(n))$ pour la diffusion ainsi qu'en temps : $O(D + \triangle \log(n))$ (où $\triangle$ est le degré maximal du graphe) pour le bavardage (échange complet d'information). La procédure pour effectuer le bavardage

commence par l'attribution à chacun des noeuds d'un rang. Ensuite, il y a le rassemblage de tous les messages en un seul message dans un seul noeud et finalement la diffusion. La diffusion utilise aussi les rangs des noeuds. Le message part de la source et suit les chemins déterminés par les rangs.

Problèmes ouverts Existe-il un algorithme polynomial qui produit un schéma de diffusion déterministe travaillant en temps $O(D+\log^2 n)$ pour n'importe quel graphe de n noeuds avec un diamètre de D. Quelle est la diffusion la plus rapide que nous sommes capables d'atteindre ?

Diffusion dans les réseaux ad-hoc

Modèle Le deuxième cas traité est celui du réseau ad-hoc, c'est-à-dire, un système complètement distribué, déployé sans infrastructure fixe. Il n'y a pas de moniteur ni de tableau de routage central. Aucune station ne connaît la topologie du réseau et celle-ci peut changer dans le temps.

Dans les réseaux ad-hoc, les noeuds ont une connaissance limitée du réseau. C'est-à-dire que les noeuds ont connaissance de leur entourage (les voisins immédiats ou même plus loin, dépendant de leur portée de connaissance). Parfois, on suppose que les noeuds ne connaissent que leur propre étiquette. Dans ce cas, il a été prouvé (dans [3]) que pour n'importe quel protocole $\prod$, n et $D \leq n/2$, il existe un réseau G avec n noeuds et un diamètre D pour lequel le nombre de rondes nécessaires pour que $\prod$ termine la diffusion d'un message dans G est $\Omega(D \log n)$.

Résultats Le premier article à traiter la diffusion déterministe avec des noeuds n'ayant pas connaissance de la topologie du graphe est : [2]. En 1997, les auteurs de [3] se sont aussi préoccupés du cas où les noeuds ne connaissent pas l'identité de leurs voisins. Cette hypothèse est motivée par le fait que dans les réseaux mobiles ou les réseaux à haute vitesse, la topologie change souvent. Cet article a donné la borne inférieure nommé ci-dessus $\Omega(D \log n)$ qui a ensuite été renforcée à $\Omega(n \log D)$ dans [9]. Par contre, cette borne est invalide si les processeurs se réveillent spontanément (c'est-à-dire qu'ils peuvent transmettre avant même d'avoir reçu un message) ou s'il y a un système de détection de collisions.

Entre 2002 et 2005, il y a eu une course à l'algorithme de diffusion le plus efficace dans les réseaux ad hoc modelisés par des graphes orientés à topologie inconnue. Un algorithme travaillant en temps $O(n^2)$ peut être obtenu très facilement. L'article [7] propose un algorithme

qui travaille en temps $O(n^{\frac{11}{6}})$. L'idée principale de l'algorithme est l'utilisation du concept combinatoire des familles sélectives qui permettent de trouver une ronde de transmission sans collision dans un temps relativement court.

Peu après, l'article [6] reprend la même idée des familles sélectives de [7] et propose un algorithme qui travaille en temps $O(n^{\frac{9}{5}})$. Au lieu de fonctionner avec seulement deux familles, il utilise n familles. Un noeud du réseau peut être dans un des trois états différents :

1. Non informé si le noeud n'a jamais reçu le message
2. Actif dans la ronde lorsque le noeud vient d'être informé du message pour la première fois
3. Passif lorsque le noeud a informé tous ses voisins

L'analyse de l'algorithme procède en comptant le progrès effectué. Lorsqu'un noeud étant le seul noeud actif diffuse, il deviendra nécessairement passif. Lorsqu'un noeud change d'état, il y a un progrès de fait. Lorsqu'un noeud passe de non informé à actif ou d'actif à passif, il y a une unité de progrès. Lorsqu'un noeud change de non informé à passif, il y a deux unités de progrès. Lorsque nous avons atteint 2n - 1 unités de progrès, tous les noeuds sont maintenant passifs et donc, la diffusion est complète.

Ensuite, l'article [8] propose un algorithme qui travaille en temps $O(n \log^2 n)$. (Il y a eu deux autres résultats intermédiaires $O(n^{\frac{5}{3}})$ $O(n^{\frac{3}{2}})$ qui ne seront pas traités dans cette revue de la littérature). Ce résultat se rapproche plus étroitement de la borne inférieure $\Omega(n \log D)$.

Les deux derniers résultats sont : $O(n \log n * \log D)$ de l'article [16] et $O(n \log^2 D)$ de [10]. Ces deux derniers résultats se rapprochent encore plus de la borne inférieure.

Le problème étudié dans l'article [19] est celui qui ressemble le plus à notre sujet de recherche. Il s'agit ici de réveiller tous les noeuds d'un réseau ad-hoc anonyme. Les noeuds n'ont pas d'étiquettes uniques ce qui augmente le niveau de difficulté. C'est le premier article qui étudie la diffusion dans les réseaux anonymes.

Au commencement, seulement la source connaît le message et à la fin tous les noeuds du réseau doivent être réveillés en entendant ce message. Un noeud est accessible s'il peut être atteint par un algorithme de diffusion déterministe possiblement dépendant du réseau. Un algorithme est universel s'il peut atteindre tous les noeuds accessibles dans tous les réseaux. Cet article répond à la question suivante : Existe-il un tel algorithme universel ?

Dans cette situation tous les noeuds sont anonymes et n'ont pas connaissance de la topologie du réseau, donc la seule façon pour un noeud de décider s'il doit transmettre ou

non, c'est de se baser sur les messages qu'il a reçus. À ce point, un algorithme de réveil doit être vu comme une fonction prenant l'histoire des communications du noeud et retournant la décision de transmettre ou non.

Cet article prouve qu'il existe un tel algorithme universel pour réveiller tous les noeuds accessibles dans les réseaux ad-hoc anonymes synchrones. Il prouve aussi que la réponse est négative pour les réseaux asynchrones.

Problèmes ouverts Existe-il un algorithme de diffusion déterministe qui permet de faire la diffusion en temps $(n \log D)$ pour n'importe quel graphe de n noeuds avec un diamètre de D ? Les algorithmes existant se rapprochent de plus en plus de la borne inférieure $\Omega(n \log D)$.

Maintenant que l'existence d'un algorithme universel pour réveiller les réseaux ad-hoc anonymes synchrones a été prouvé, il faut trouver des algorithmes efficaces pour cette tâche. L'algorithme présenté dans [19] est exponentiel.

2.1.2 Algorithmes aléatoires

Modèle

Le troisième cas traité est celui du réseau radio avec des algorithmes aléatoires. Ces algorithmes peuvent être utilisés dans les différentes structures de réseaux nommées précédemment (c'est-à-dire, les réseaux centralisés et les réseaux ad-hoc). Avec de tels algorithmes, il n'est pas nécessaire d'utiliser les étiquettes uniques.

Ce que nous cherchons à obtenir c'est un algorithme qui fonctionne avec une très grande probabilité. Ici, le problème des noeuds non accessibles (pour cause de symétrie du réseau) ne s'applique plus. Si nous envoyons, par exemple, 10 000 fois le message avec une probabilité $\frac{1}{2}$, il y aura au moins une transmission sans collision avec très forte probabilité.

Pour une marge d'erreur ϵ donnée, on cherche des algorithmes de diffusion les plus efficaces possibles qui sont corrects avec probabilité au moins $1 - \epsilon$.

Résultats

Le premier algorithme de diffusion aléatoire a été donné dans [2]. L'idée générale de cet algorithme est d'éliminer la moitié des processeurs à chaque tour, en espérant que pendant un tour, il n'y aura qu'un transmetteur. La procédure d'affaiblissement utilisée dans cet

algorithme dit aux noeuds d'effectuer une transmission tant qu'ils sont actifs. À chaque tour les noeuds ont 50% des chances de devenir inactifs et 50% des chances de rester actifs.

Cet algorithme nous donne un temps de diffusion de $O((D + \log n/\epsilon) \log n)$. La diffusion peut donc s'effectuer en $O(D \log n + \log^2 N)$ rondes.

En 1998, les auteurs de [17] ont prouvé que la borne inférieure pour le temps moyen d'une diffusion aléatoire était $\Omega(D \log(n/D))$. Cinq ans plus tard, un algorithme optimal [10] qui fonctionne en $O(D \log(n/D) + \log^2 N)$ rondes a été proposé. Cet algorithme est une amélioration de [2].

2.2 La technologie des réseaux sans fil

Les réseaux sans fil permettent de connecter les ordinateurs sans se soucier de leur emplacement ou des câbles. Les appareils communiquent par des ondes radio (ondes hertziennes) dans une zone de couverture définie. Les ondes hertziennes sont des ondes électromagnétiques, leurs longueurs d'onde et leurs fréquences permettent de coder toutes sortes d'informations. Elles se propagent dans l'air entre les antennes.

Avec l'intérêt pour la communication sans fil qui se développe de jours en jours, plusieurs technologies de réseaux sans fil sont actuellement en développement sur le marché. Il y a beaucoup de différences dans leurs caractéristiques techniques car elles sont destinées à différents usages. Dans ce qui suit, nous allons faire une brève comparaison entre ces différentes technologies.

2.2.1 Les catégories de réseaux sans fil

Réseaux personnels sans fil (WPAN)

Le réseau personnel sans fil (WPAN pour Wireless Personal Area Network) ce sont des réseaux sans fil d'une faible portée : quelques dizaines mètres. Ces réseaux servent à relier des périphériques (imprimante, cellulaires) ou un assistant personnel (PDA) à un ordinateur. Ils peuvent aussi permettre la liaison sans fil entre deux machines très peu distantes. [11]

Voici les différentes technologies utilisées pour les WPAN :

La technologie Bluetooth lancée par Ericsson en 1994, proposant un débit théorique de 1 mégabit par seconde pour une portée maximale d'une trentaine de mètres et elle est très peu gourmande en énergie.

La technologie HomeRF (pour Home Radio Frequency) apparue en 1998 par le HomeRF Working Group propose un débit théorique de 10 mégabits par seconde avec une portée d'environ 50 à 100 mètres sans amplificateur.

La technologie ZigBee permet d'obtenir des liaisons sans fil à très bas prix et avec une très faible consommation d'énergie, elle est directement intégrée dans de petits appareils électroniques (appareils électroménagers, hifi, jouets).

La technologie des liaisons infrarouges permet de créer des liaisons sans fil de quelques mètres avec des débits pouvant monter à quelques mégabits par seconde. Cette technologie est largement utilisée pour la domotique (télécommandes), mais il y a des perturbations dues aux interférences lumineuses.

Réseaux locaux sans fil (WLAN)

Le réseau local sans fil (WLAN pour Wireless Local Area Network) est un réseau permettant de couvrir l'équivalent d'un réseau local d'entreprise, soit une portée d'environ une centaine de mètres. Il permet de relier entre-eux les terminaux présents dans la zone de couverture. Voici les deux principales technologies utilisées :

Le Wifi soutenu par l'alliance WECA (Wireless Ethernet Compatibility Alliance) offre des débits allant jusqu'à 54Mbps sur une distance de plusieurs centaines de mètres.

HiperLAN2 (High Performance Radio LAN 2.0) norme européenne élaborée par l'ETSI (European Telecommunications Standards Institute). HiperLAN 2 permet d'obtenir un débit théorique de 54 mégabits par seconde sur une zone d'une centaine de mètres dans la gamme de fréquence comprise entre 5 150 et 5 300 mégahertz.

Réseaux métropolitains sans fil (WMAN)

Le réseau métropolitain sans fil (WMAN pour Wireless Metropolitan Area Network) est connu sous le nom de Boucle Locale Radio (BLR). Les WMAN sont basés sur la norme IEEE 802.16. La boucle locale radio offre un débit utile de 1 à 10 Mbit/s pour une portée de 4 à 10 kilomètres, ce qui destine principalement cette technologie aux opérateurs de télécommunications.

Réseaux étendus sans fil (WWAN)

Le réseau étendu sans fil (WWAN pour Wireless Wide Area Network) est également connu sous le nom de réseau cellulaire mobile. Il s'agit des réseaux sans fil les plus répandus puisque tous les téléphones mobiles sont connectés à un réseau étendu sans fil.

2.2.2 Les réseaux ad hoc

Pourquoi les réseaux ad-hoc sont-ils importants ? Parce que le déploiement d'une infrastructure fixe peut être très coûteuse ou même parfois impossible. Bien que les réseaux ad-hoc aient une portée limitée, un utilisateur peut aisément faire une communication au-delà de sa portée. Chaque utilisateur du réseau peut retransmettre les messages afin que tous les utilisateurs puissent joindre tous les autres, quelle que soit la distance qui les sépare, pourvu qu'il y ait assez d'utilisateurs répartis sur le chemin. [18] Un réseau ad-hoc est donc autonome et supporté par la collaboration de l'ensemble des ses participants.

Une interface de communication radio permet aux utilisateurs d'accéder au réseau facilement tout en pouvant être mobile. Néanmoins, cet immense avantage souffre d'inconvénients importants qui limitent le débit disponible pour l'utilisateur. L'utilisation de l'air comme support de communication est soumise à des réglementations strictes qui empêchent d'utiliser n'importe quelle fréquence. Il est donc impossible d'augmenter à volonté le débit afin de répondre à toutes les demandes. De plus, un problème majeur est celui du partage de l'accès au médium. En effet, si deux utilisateurs veulent parler en même temps sur le même médium (soit l'air), il se produit une collision. Dans le meilleur des cas, une seule des deux communications va être utilisable. En général, les deux communications seront brouillées et la communication aura été faite pour rien, ce qui diminue le débit (car on a perdu du temps) et dans le cas des périphériques portables, consomme inutilement de l'énergie. Voilà pourquoi il est très important de chercher des techniques pour diminuer les collisions.

2.3 Comparaison des résultats connus avec le problème du mémoire

Comme nous pouvons le constater, il y a beaucoup de travail qui a été fait sur les réseaux radio. Nous avons regardé certains des algorithmes probabilistes et déterministes.

Le sujet de ce mémoire concerne seulement les algorithmes déterministes. Nous cherchons à trouver un algorithme déterministe qui informe tous les noeuds accessibles.

Un autre fait à noter est que dans les recherches antérieures lorsqu'il y avait des algorithmes déterministes pour la diffusion dans un réseau radio, les noeuds possédaient toujours des étiquettes uniques (sauf [19]).

Dans notre recherche, nous ne supposons pas l'existence d'étiquettes uniques. La plupart des techniques qui ont été résumées ne s'appliqueront pas directement à notre problème. Par contre, certaines idées générales présentées dans les algorithmes connus dans la littérature pourront toujours aider à comprendre ce problème et à élaborer sa solution.

L'article [19] est le seul à utiliser exactement le même modèle que nous et à proposer à un algorithme de diffusion pour les noeuds anonymes. Par contre, cet article s'intéresse plutôt à l'existence d'un tel algorithme dans n'importe quel réseau radio sans souci d'efficacité. En fait, l'algorithme de [19] est exponentiel. En revanche, dans notre mémoire nous cherchons à trouver un tel algorithme pour une classe particulière de réseaux (réseaux à topologie de grilles rectangulaires), mais nous construisons un algorithme beaucoup plus efficace que l'algorithme universel de [19]. Nous justifions cette assertion par des simulations.

Chapitre 3

Présentation des résultats de la recherche

Dans ce chapitre, nous allons présenter les résultats de notre recherche sur la diffusion de messages dans les réseaux radio anonymes.

Nous présenterons certains résultats de la diffusion dans quelques grilles particulières puis nous démontrerons pourquoi certains noeuds seront toujours inaccessibles peu importe l'algorithme déterministe utilisé. Dans le reste de ce mémoire, nous ne considérons que les algorithmes déterministes.

3.1 Introduction

Tel qu'expliqué précédemment, lorsque nous effectuons une diffusion dans les réseaux radio anonymes, certains noeuds sont inaccessibles. Ces noeuds se situent sur les diagonales de la grille lorsque nous sommes en présence d'une symétrie impossible à briser. Dans la section qui suit, nous allons prouver que ces noeuds sont toujours inaccessibles et discuter de l'importance de ce résultat dans notre recherche. Nous terminerons en présentant brièvement notre algorithme.

Les définitions :

- Considérons un graphe non-orienté $G = (V,E)$. Les éléments de V sont appelés les sommets et les éléments de E sont appelés les arêtes du graphe G.
- Soit $N(v) = \{u \in V : \{u, v\} \in E\}$. $N(v)$ s'appelle le voisinage du noeud v.
- Soit s : un noeud distingué du graphe G qui s'appelle la source.

3.2 L'impossibilité d'atteindre des noeuds spéciaux dans un réseau radio anonyme

Nous établirons une condition suffisante pour qu'un noeud du réseau radio anonyme représenté par le graphe G soit impossible à informer par n'importe quel algorithme.

Considérons un noeud v. Si chacun des noeuds du voisinage $N(v)$ a un noeud structurellement semblable dans ce voisinage alors peu importe l'algorithme, à chaque ronde, le noeud u et son semblable effectueront toujours la même action. Le noeud v ne pourra donc jamais être informé parce que lorsqu'un de ses voisins essaiera de l'informer, le noeud semblable le fera lui aussi, provocant ainsi une collision. Ci-dessous nous précisons ces notions.

Automorphisme d'un graphe

Un automorphisme d'un graphe est une bijection f : V $\longrightarrow$ V telle que

- $\{(f(v), f(w)\} \in E \Leftrightarrow \{v, w\} \in E$.

Un noeud a' est appelé semblable au noeud a ($a' \sim a$), s'il existe un automorphisme f du graphe G tel que :

- f(s) = s
- f(a) = a'

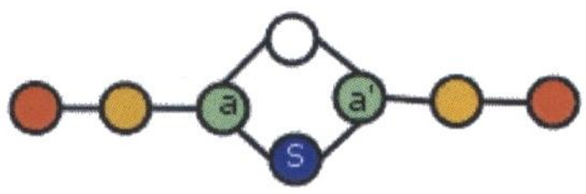

FIG. 3.1 – Graphe avec un automorphisme autre que l'identité

Histoire d'un noeud :

Nous supposons que tous les noeuds sont anonymes et n'ont pas connaissance de la topologie du réseau. La seule façon pour un noeud de décider s'il doit transmettre ou non, c'est de se baser sur les messages qu'il a reçus précédemment (l'histoire). Un algorithme de diffusion peut être vu comme une fonction prenant l'histoire des communications du noeud et retournant la décision de transmettre ou non. Lorsqu'un noeud transmet un message, nous pouvons supposer que ce message contient toute l'histoire du noeud. Tel qu'expliqué précédemment, l'histoire est la seule information sur laquelle peut se baser l'algorithme pour prendre une décision. Nous pouvons simuler n'importe quel algorithme de diffusion par un algorithme de diffusion qui transmet les histoires entières comme messages. Alors si un noeud est inaccessible par l'algorithme qui transmet les histoires, il le sera aussi par n'importe quel algorithme.

La définition formelle de l'histoire d'un noeud est : L'histoire d'un noeud v durant la ronde i appelé *H(i,v)* est une séquence finie d'entiers positifs et des symboles [ ]. Cette séquence correspond à ce que le noeud v a entendu jusqu'à la ronde i. On suppose qu'au début le noeud connaît seulement son degré et sait s'il est ou non la source de la diffusion.

Voici la façon de définir l'histoire d'un noeud : $H(0,v) = [1, d]$ si v est la source, sinon $H(0,v) = [d]$, où d est le degré de v.

Supposons que *H(j,w)* est défini pour tous les $j < i$ ainsi que pour tous les noeuds w. Considérons un noeud v durant la ronde $i > 0$.

- Si le noeud v a reçu un message m durant la ronde i alors *H(i,v)* = concaténation de *H(i-1,v)* et la séquence [m].
- Si le noeud v n'a pas entendu le message m durant la ronde i alors *H(i,v)* = concaténation de *H(i-1,v)* et la séquence [ ].
- Si le noeud v n'a encore jamais entendu un message pendant les rondes $j < i$ alors $H(i,v) = H(0,v)$.

$\mathcal{H}$ est définit comme l'ensemble de toutes les histoires. Un algorithme de diffusion A est une fonction $A : \mathcal{H} \longrightarrow \{0, 1\}$, tel que $A[d] = 0$, pour chaque entier d (intuitivement, les noeuds autres que la source ne transmettent pas avant la réception d'un message). Lorsque $A(H) = 1$, l'algorithme A ordonne au noeud avec une histoire H d'envoyer le message contenant H. Lorsque $A(H) = 0$, l'algorithme A ordonne au noeud avec une histoire H de se taire. Nous définissons par *J(v,i,A)* l'histoire du noeud v après la ronde i de l'exécution de l'algorithme A.

Lemme 1. 1. *Soit (G,s) un réseau radio anonyme. Si* a' $\sim$ a, *alors pour chaque ronde* i *et chaque algorithme A, nous avons :*

$$(*)\ \mathrm{J}\ \ (\mathrm{a},\ \mathrm{i},\ \mathrm{A})\ =\ \mathrm{J}\ \ (\mathrm{a'},\ \mathrm{i},\ \mathrm{A})$$

Démonstration :

Notons d'abord que si $a' \sim a$, alors les degrés de a' et de a sont égaux. Nous démontrons la propriété (*) par induction sur i.

La propriété est vraie pour $i = 0$ car pour chaque algorithme A et pour tous les noeuds v et w différents de s et tels que les degrés de v et w sont égaux on a :

$$J\ \ (v,\ 0,\ A)\ =\ J\ \ (w,\ 0,\ A)$$

Supposons que (*) est vraie pour i - 1 avec n'importe quel algorithme et soit $a' \sim a$. Donc $J\ \ (a,\ i\text{-}1,\ A)\ =\ J\ \ (a',\ i\text{-}1,\ A)$. Si aucun des noeuds a ou a' n'entend un message pendant la ronde i, alors (*) est vraie pour i. Supposons donc que a entend un message pendant la ronde i et soit $v \in N(a)$ le noeud qui a transmis ce message.

Soit f l'automorphisme de G tel que $f(s) =$ s et que $f(a) =$ a'. Alors $f(v) \in N(f(a'))$. Le même automorphisme f montre que v et $f(v)$ sont semblables.

Par l'hypothèse d'induction :

$$J\ \ (v,\ i\text{-}1,\ A)\ =\ J\ \ (f(v),\ i\text{-}1,\ A)$$

Donc, $f(v)$ transmet le même message que v en suivant l'algorithme A. La question est : est-ce que a' entend ce message ? Supposons que non. Il y a deux cas possibles, soit a' transmet à lui-même dans la ronde i, soit il existe un noeud $w \in N(a')$ tel que $w \neq f(v)$ et w transmet dans la ronde i. Cependant, $f^{-}1(w) \sim w\ et\ a' \sim a$. Donc, par l'hypothèse d'induction :

$$J\ \ (\mathrm{a},\ i\text{-}1,\ A)\ =\ J\ \ (\mathrm{a'},\ i\text{-}1,\ A)\ \ et\ \ J\ \ (f^{-1}(w),\ i\text{-}1,\ A)\ =\ J\ \ (w,\ i\text{-}1,\ A)$$

Dans le premier cas, le noeud a transmet pendant la ronde i et dans le deuxième cas le noeud $f^{-}1(w)$ ainsi que le noeud v transmettent pendant cette ronde. Dans les deux cas, a ne peut pas entendre le message ce qui donne une contradiction. Cette contradiction montre que pendant la *i-ième* ronde a' entend le message de $f(v)$. Ce message étant identique au

message que v transmet pendant cette ronde, on a J (a, i, A) = J (a', i, A). Ce qui prouve le lemme par induction. ■

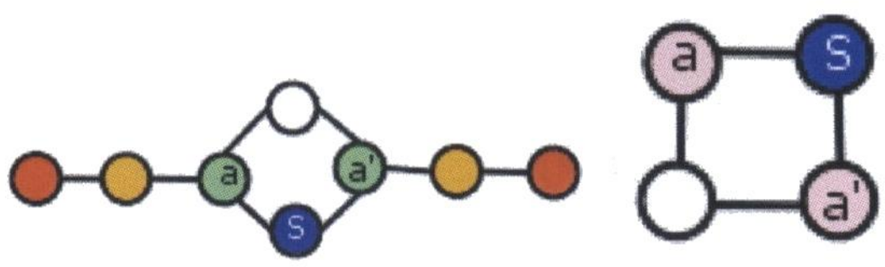

FIG. 3.2 – Exemples de noeuds semblables a et a'

Un ensemble S de noeuds est appelé spécial si chaque $v \in S$ a la propriété suivante : $\forall\ a \in N(v) \setminus S,\ \exists\ a' \in N(v)\ (a' \sim a \wedge a \neq a')$. Un noeud est spécial s'il est élément d'un ensemble spécial.

Théorème 1. 1. *Aucun noeud spécial ne peut être informé par aucun algorithme.*

Démonstration :

Soit S, un ensemble spécial. Nous montrerons qu'aucun élément de S ne peut être informé. Soit A, un algorithme quelconque et soit w, le premier noeud de S informé par A.

Supposons que $a \in N(w) \setminus S$ est le noeud qui informe w. Supposons que la communication a eu lieu pendant la ronde i. Soit $a' \in N(w)$ tel que $a' \sim a$ et $a' \neq a$. En vue du lemme 1.1, on a :

$$J\ (\text{a},\ i\text{-}1,\ A) = J\ (\text{a'},\ i\text{-}1,\ A)$$

Donc, si a transmet pendant la ronde i, alors a' le fait aussi. Ce qui implique que w ne peut pas entendre le message pendant cette ronde. Nous avons donc une contradiction, ce qui prouve que tous les noeuds spéciaux sont inaccessibles. ■

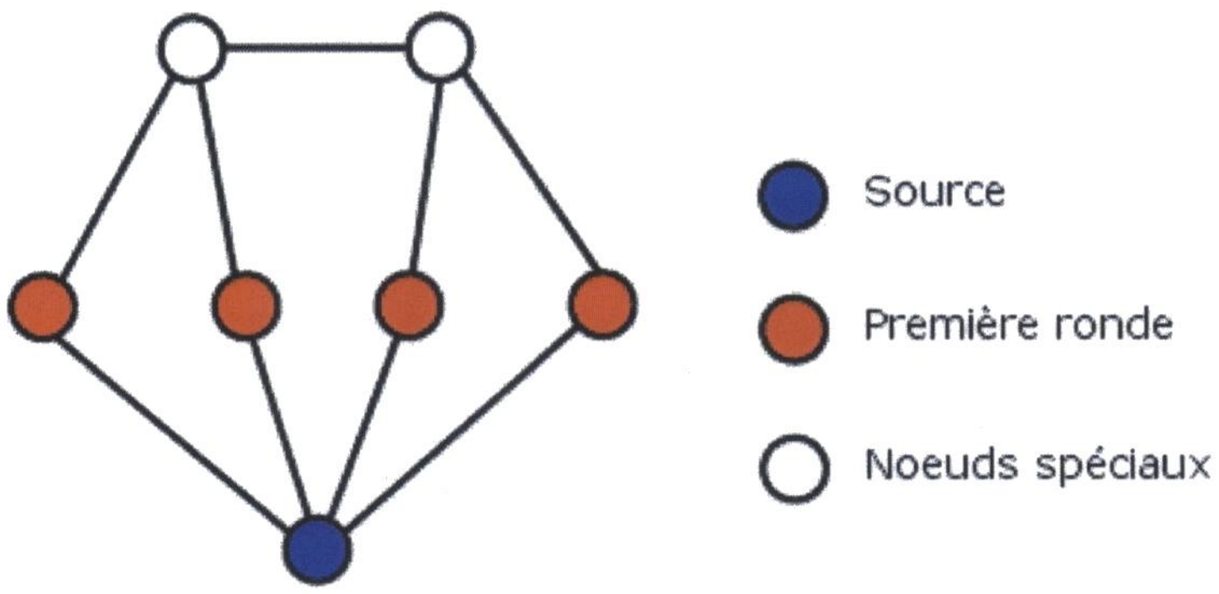

FIG. 3.3 – Exemple d'un ensemble spécial

Dans le cas spécifique des grilles, il existe deux situations dans lesquelles nous avons des ensembles de noeuds spéciaux. La première situation est celle d'une grille carrée impaire avec la source qui est située au centre. L'ensemble spécial est constitué alors de tous les noeuds sur les deux diagonales en excluant la source (voir figure 3.4). La deuxième situation est celle où la source est située sur une seule diagonale d'une grille carrée. L'ensemble spécial est constitué alors de tous les noeuds sur cette diagonale sauf la source (voir figures 3.5 et 3.6).

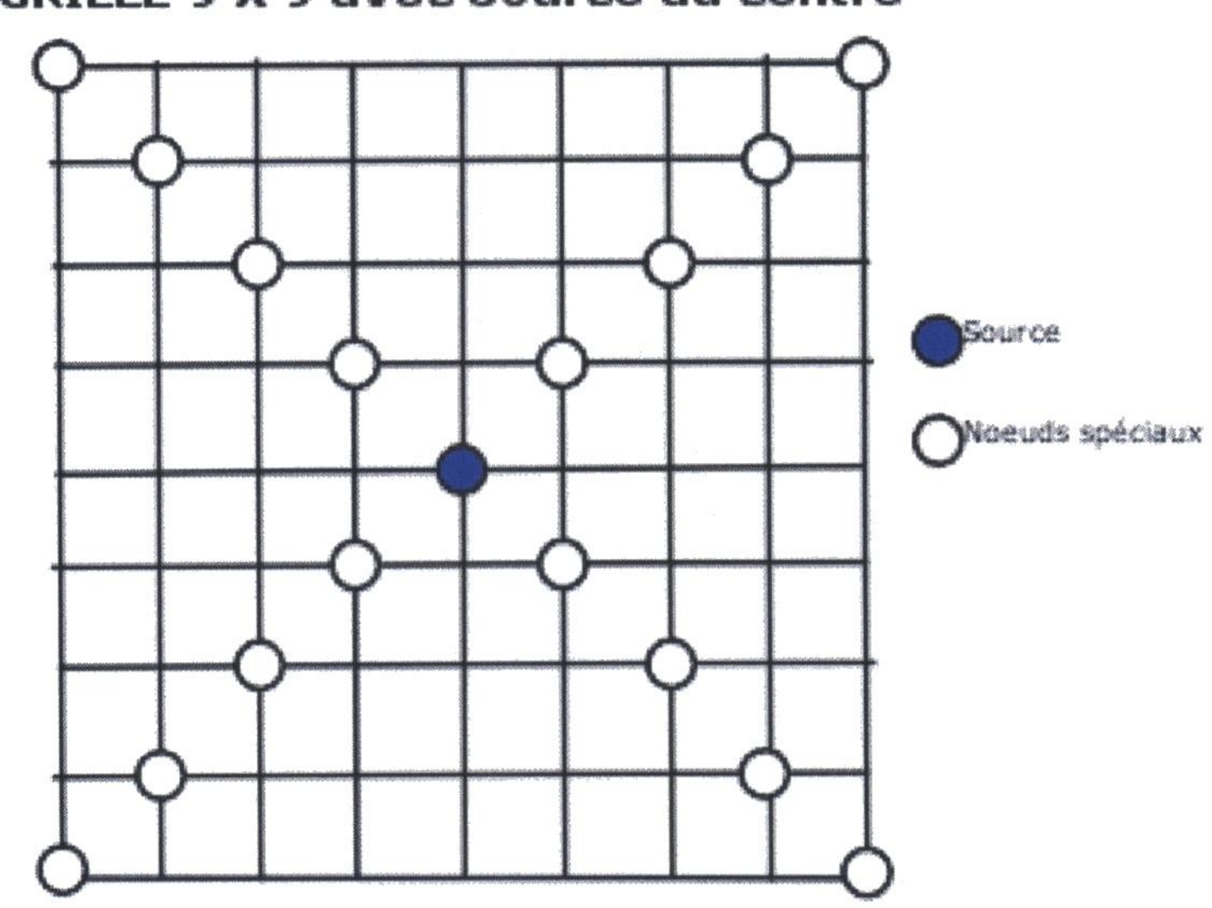

FIG. 3.4 – Ensemble spécial dans une grille carrée impaire avec la source centrée

3.3 L'algorithme informant les noeuds accessibles

Suite à notre résultat d'impossibilité, nous savons maintenant que les noeuds spéciaux seront toujours inaccessibles. Qu'en est-t-il des autres noeuds d'une grille rectangulaire donnée ? Nous formulons l'hypothèse qu'il est possible d'atteindre tous ces autres noeuds dans chaque grille rectangulaire. Notre objectif est de trouver un tel algorithme.

Afin de créer cet algorithme nous avons détaillé toutes les positions possibles de la source et cherché une procédure de diffusion dans chacun de ces cas. Ensuite, à partir de ces procédures, nous avons construits un algorithme général de diffusion qui fonctionne même quand la position de la source est inconnue.

Cet algorithme nous permet d'informer tous les noeuds qui ne sont pas inaccessibles et nous parvenons à le faire malgré l'anonymat des noeuds. Tel qu'expliqué précédemment, le fait que les noeuds soient anonymes apporte une difficulté supplémentaire. Afin de surmonter ce problème, il faut utiliser plusieurs astuces. Donc, à première vue notre algorithme peut

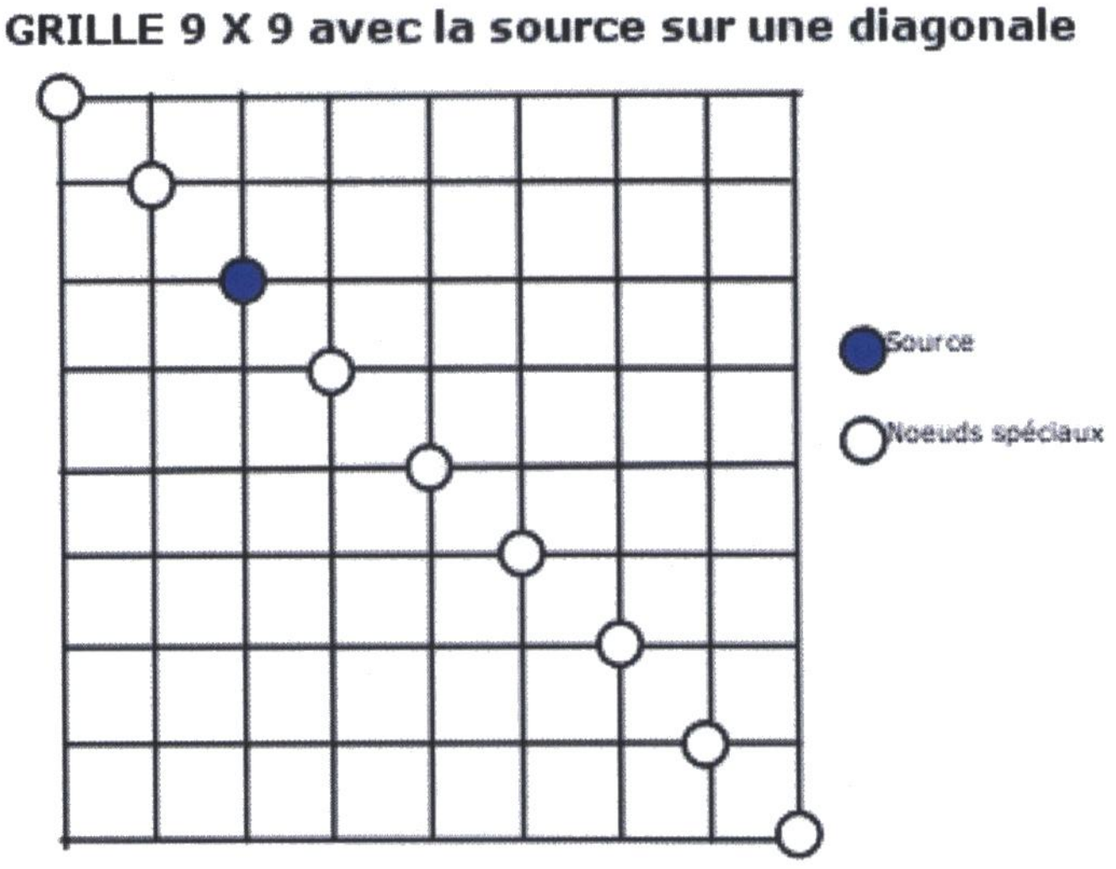

FIG. 3.5 – Ensemble spécial dans une grille carrée impaire avec la source située sur une diagonale

sembler beaucoup trop complexe pour effectuer une simple diffusion dans une grille rectangulaire. Par contre, dû à l'anonymat, il est impossible de demander seulement à certains noeuds directement de transmettre. Les procédures utilisées étaient donc nécessaires pour remplir notre objectif. Nous présenterons cet algorithme dans la prochaine section.

Nous procéderons ensuite aux simulations de notre algorithme qui justifient de façon expérimentale son exactitude et montrent son efficacité en pratique.

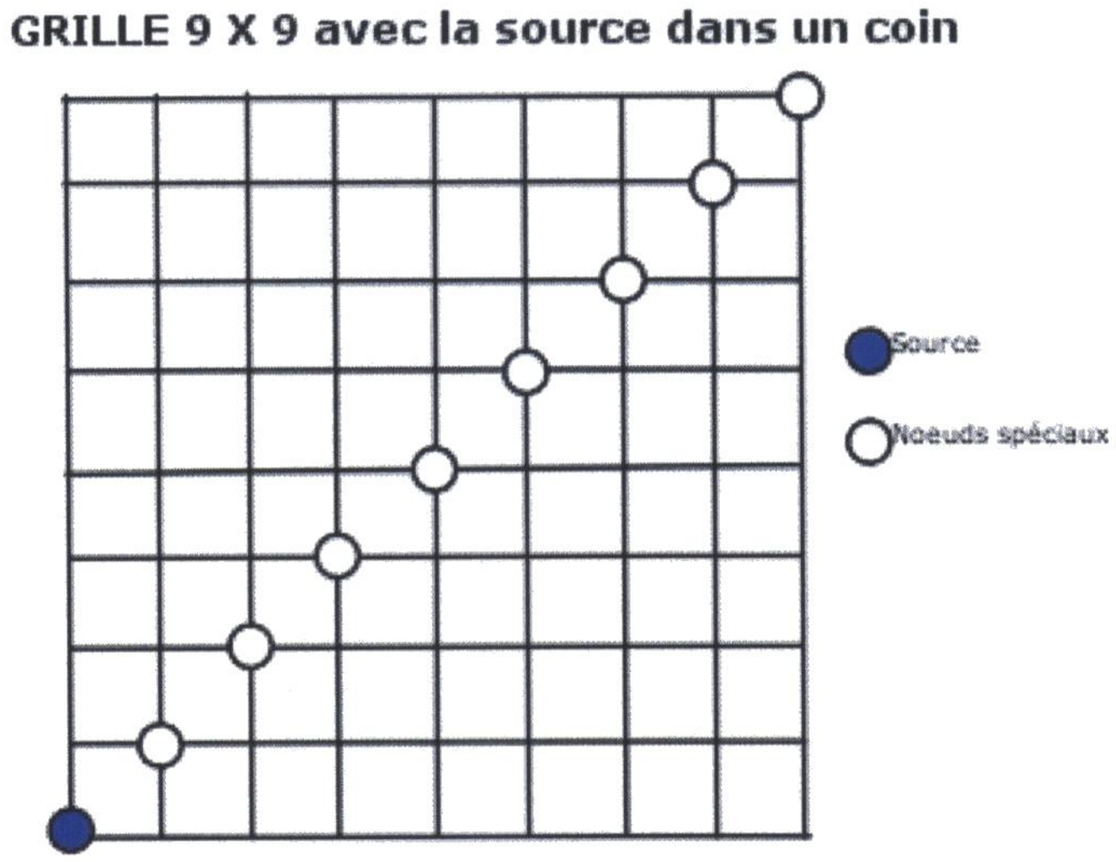

FIG. 3.6 – Ensemble spécial dans une grille carrée avec la source située dans un coin

Chapitre 4

Algorithme de diffusion de messages dans les réseaux radio anonymes de type grille rectangulaire

4.1 Introduction

Ce chapitre décrit notre algorithme de diffusion pour tous les réseaux radio anonymes sous forme de grilles. Ces réseaux ne sont pas orientés et doivent être une grille rectangulaire n par m où n et m sont supérieurs ou égaux à 2 et n peut être égal à m. Dans certains cas, il y a des noeuds qui ne pourront jamais être informés. Ces noeuds sont des noeuds inaccessibles comme il a été prouvé dans la preuve d'impossibilité du chapitre 3.

Avant de présenter l'algorithme, il faut expliquer pourquoi nous devions créer un algorithme aussi complexe. En regardant rapidement le problème, nous pourrions croire qu'une solution facile pourrait très bien résoudre le problème. Malheureusement, ce n'est pas le cas, nous allons donc regarder quelques exemples illustrant pourquoi cela ne pourrait pas fonctionner.

Premièrement, essayons l'algorithme de diffusion le plus simple qui existe. La source transmet le message à ses voisins, lorsqu'un noeud reçoit un message durant une ronde, il le retransmet pendant la prochaine ronde. Dans le cas d'une grille 3 par 3 avec une source de degrés 3, avec l'algorithme de diffusion simple, il y aura deux noeuds sur neuf qui ne seront

jamais informés. Tandis qu'avec notre algorithme, tous les noeuds seront informés dans ce cas.

L'idée principale de notre algorithme est la suivante. En premier lieux, nous procédons à l'élection d'un axe, c'est-à-dire une ligne ou une colonne comprenant la source. Ensuite, nous effectuons la synchronisation de l'axe élu : tous les noeuds de cet axe déterminent une ronde commune pour débuter la dernière partie de la diffusion. Celle-ci consiste en propagation simultanée des messages (appelée "balayage") qui informe tous les noeuds accessibles de la grille malgré les collisions. Parfois (par exemple, lorsque la source est sur une diagonale d'une grille carrée), nous ne pouvons pas faire l'élection d'un axe à cause de la symétrie. Alors nous procédons au balayage à partir d'une ligne et d'une colonne formant le contour de la grille en laissant la diagonale non informée (elle constitue l'ensemble des noeuds inaccessibles). Dans le cas spécial de la source au centre d'une grille carrée impaire, le balayage se fait simultanément à partir des deux axes en laissant, bien sûr, les deux diagonales non informées (elles constituent l'ensemble des noeuds inaccessibles).

Cet aperçu de l'algorithme suppose que la position de la source est connue. Cependant l'algorithme fonctionne aussi sans cette information. Afin de compléter la diffusion nous avons décrit la procédure de diffusion pour les huit différents cas de la position de la source qui couvrent toutes les possibilités. Lorsque la position de la source est inconnue, nous effectuons le multiplexage de ces huit procédures : pendant la ronde *8(k-1) + r*, nous exécutons le k-ième pas de la procédure r $=\{1, 2, ..., 8\}$. De cette façon la diffusion sera accomplie par la procédure correspondante au cas courant (malgré la position de la source inconnue).

Ensuite, nous montrerons à l'aide des simulations que les seuls noeuds que nous ne pouvons jamais atteindre sont ceux qui sont situés sur une ou deux diagonale(s) d'une grille carrée lorsque la source est sur cette (ces) diagonale(s). Ces noeuds inaccessibles sont les noeuds constituant les ensembles spéciaux. Voici un rappel des deux situations comprenant un ensemble spécial : une grille carrée impaire avec une source centrée (l'ensemble spécial est constitué des noeuds des deux diagonales) et une grille carrée avec la source située sur une seule diagonale (l'ensemble spécial est constitué des noeuds de cette diagonale).

Lorsqu'il y a une symétrie entre deux noeuds, il y aura une collision lorsque ces noeuds transmettront. C'est pourquoi nous cherchons à briser ces symétries. Par contre, dans la diffusion simple, il restera souvent beaucoup de noeuds qui ne sont pas inaccessibles mais qui ne seront pas atteints.

Notre algorithme permet, même si les noeuds sont sans etiquette, de sélectionner indirectement certains noeuds et de s'en servir pour faire la diffusion. Ce procédé est assez complexe et il fallait bien différencier et identifier tous les cas possibles afin de construire cet algorithme. C'est pourquoi notre algorithme peut sembler long et complexe pour effectuer une diffusion dans une grille rectangulaire, mais en fait, il permet de maximiser le nombre de noeuds atteints.

Le chapitre est subdivisé comme suit : présentation des notions des axes et demi-axes (section 4.2), explication des différents cas possibles (section 4.3) et brève déclaration de la notation et terminologie utilisées (section 4.4). Par la suite, présentation de l'algorithme (section 4.5) et explication des résultats de cet algorithme (section 4.6).

4.2 Les axes et les demi-axes

1. Les axes : ce sont la ligne et la colonne qui ont comme point de rencontre la source.
   (a) L'axe est interne lorsque ses noeuds (autres que les extrémités) sont de degré 4 (voir la figure 4.1)
   (b) L'axe est externe lorsque ses noeuds (autres que les extrémités) sont de degré 3 (voir la figure 4.2)
2. Les demi-axes : Lorsque la source divise un axe en deux parties, ces deux parties sont aussi appelées les demi-axes. (voir la figure 4.3)

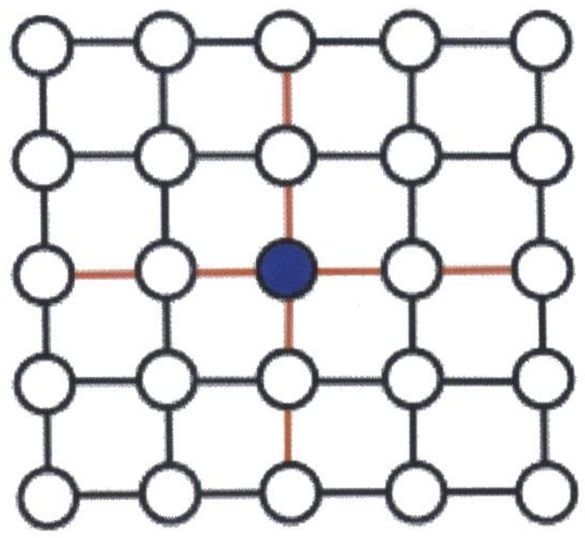

FIG. 4.1 – Les axes internes d'une grille rectangulaire

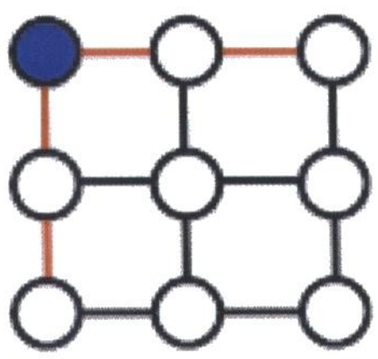

FIG. 4.2 – Les axes externes d'une grille rectangulaire

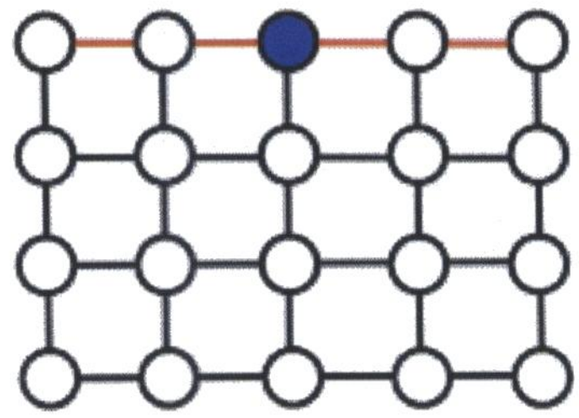

FIG. 4.3 – Les demi-axes d'une grille rectangulaire

4.3 Les différents cas possibles

1. La source est dans un coin de la grille (degré 2) :

   (a) Les deux axes sont de longueurs différentes (voir la figure 4.4)

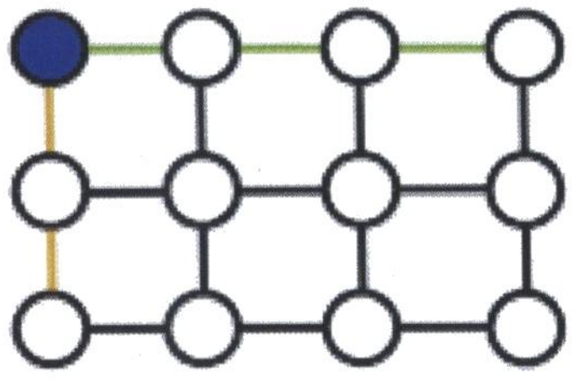

FIG. 4.4 – Deux axes de longueur différente

   (b) Les deux axes sont de même longueur (voir la figure 4.5)

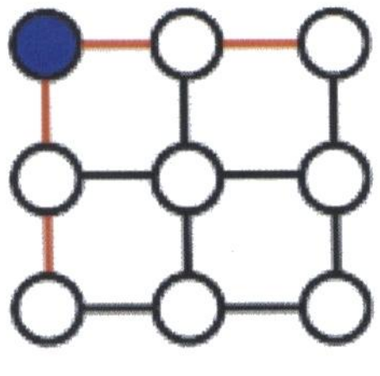

FIG. 4.5 – Deux axes de même longueur

2. La source est sur un côté de la grille (degré 3) :

   (a) Les deux demi-axes sont de longueurs différentes (voir la figure 4.6)

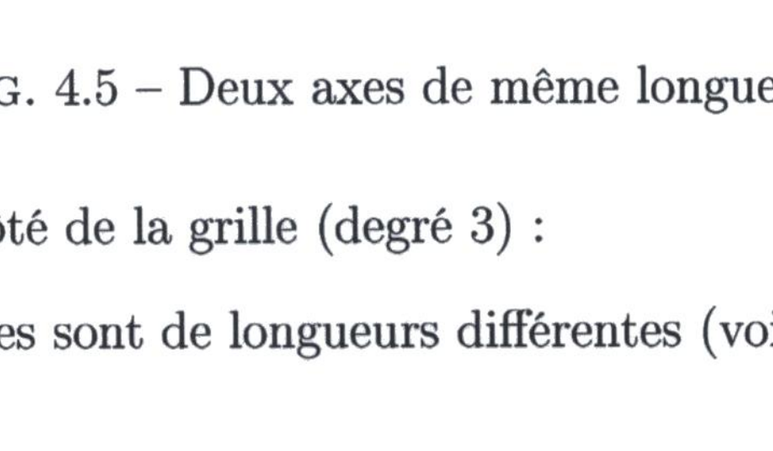

FIG. 4.6 – Deux demi-axes de longueur différente

   (b) Les deux demi-axes sont de même longueur (voir la figure 4.7)

3. La source est de degré 4 :

   (a) Les 4 demi-axes sont égaux (voir la figure 4.8)

   (b) Les demi-axes sont égaux 2 à 2

      i. Les demi-axes égaux sont adjacents - égaux en coin (voir la figure 4.9)

      ii. Les demi-axes égaux sont opposés (voir la figure 4.10)

   (c) Il y a au moins un demi-axe qui est de longueur unique (voir la figure 4.11)

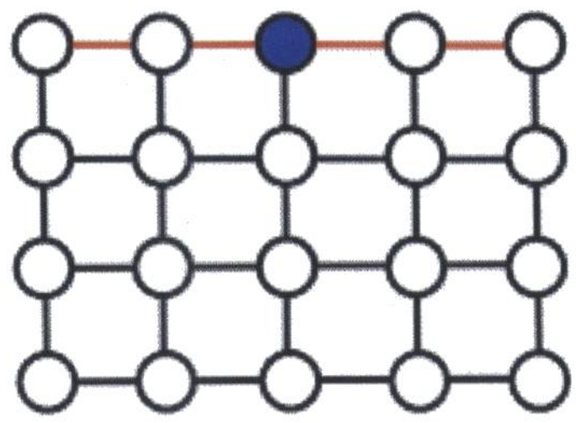

FIG. 4.7 – Deux demi-axes de même longueur

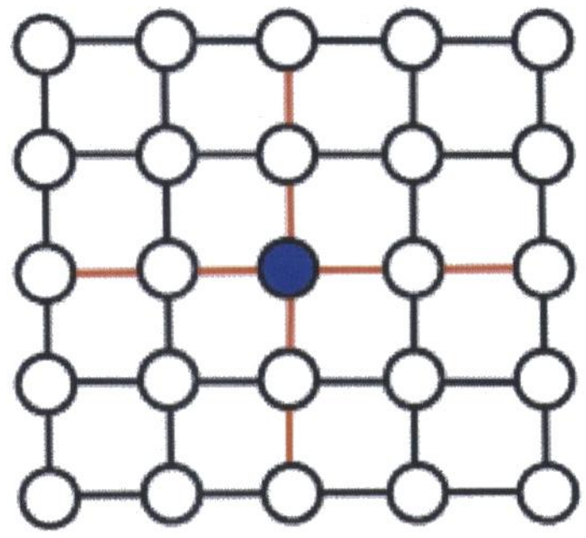

FIG. 4.8 – Quatre demi-axes de même longueur

4.4 La notation et terminologie utilisées

1. Source : le noeud qui débute la diffusion.
   (a) le noeud qui est la source : noeud(source) == vrai
   (b) tous les autres noeuds : noeud(source) == faux
2. Compteur : est le compteur utilisé pour donner indirectement des numéros aux noeuds. Plusieurs noeuds peuvent avoir le même numéro. compteur = 1 (initialisation du compteur), compteur ++ (incrémentation du compteur)
3. Les propriétés d'un noeud :
   (a) Noeud(degré) == Z, le noeud est de degré Z

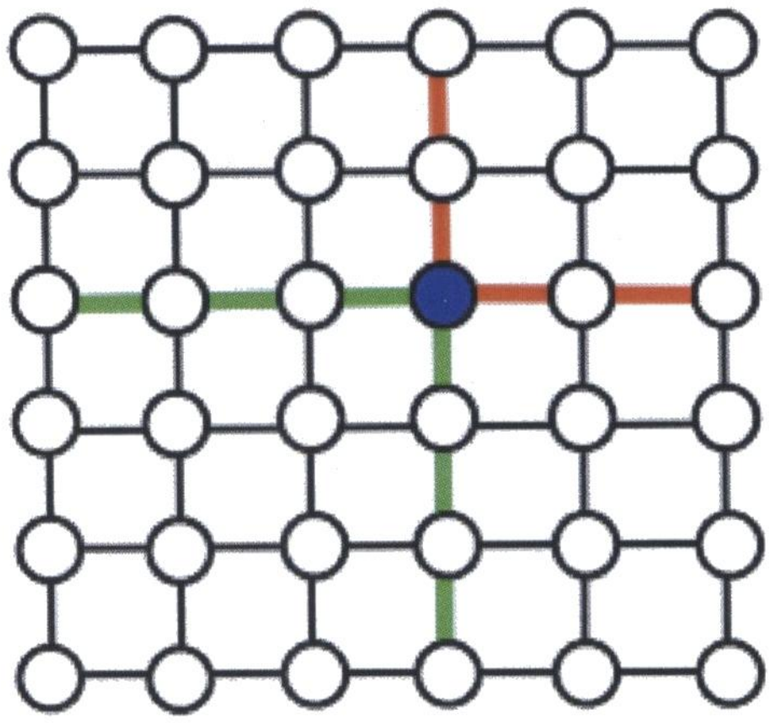

FIG. 4.9 – Les demi-axes égaux sont adjacents - égaux en coin

FIG. 4.10 – Les demi-axes égaux sont opposés - égaux en face à face

(b) Noeud(numéro) == X, le numéro du noeud est X (ronde durant laquelle il a été informé)

(c) Noeud(informé) == vrai, lorsque le noeud a déjà été informé du message, faux sinon. Au commencement, tous les noeuds ont la valeur faux, sauf la source.

(d) Noeud(élu) == vrai, si le noeud est élu, faux sinon. (Les noeuds élus seront utilisés pour effectuer la diffusion par balayage dans la grille)

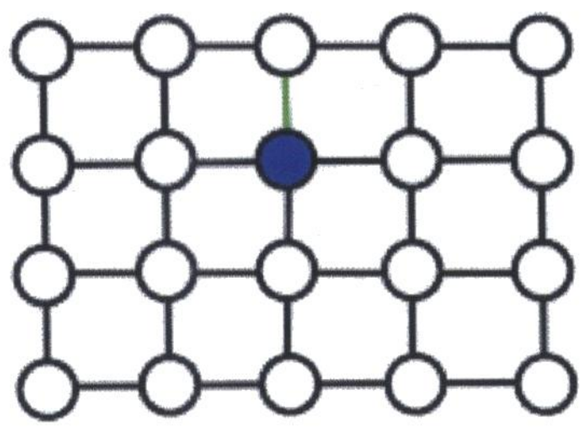

FIG. 4.11 – Au moins un demi-axe de longueur unique

(e) Noeud(chef ligne) == vrai, si le noeud est un chef de la ligne, faux sinon. (un chef de ligne est le dernier noeud informé sur une ligne ou une partie de la ligne, dans la grille)

(f) Noeud(chef spécial) == vrai, si le noeud est un chef spécial, faux sinon. (Un chef spécial est un noeud dans un coin qui servira à départager deux côtés de la grille.) Il doit conserver :

   i. Noeud(numéro X) == XX lorsque le noeud reçoit un premier message contenant le numéro XX d'un chef.

   ii. Noeud(numéro Y) == YY lorsque le noeud reçoit un seconde message contenant le numéro YY d'un autre chef.

(g) Noeud(plus petit champion) == vrai, si le noeud est le plus petit champion, faux sinon. (Le noeud plus petit champion est un noeud élu champion qui servira à synchroniser le balayage dans certains cas. Entre deux numéros reçus par la source ou par un chef, son numéro est le plus petit.)

(h) Noeud(plus grand champion) == vrai, si le noeud est le plus grand champion, faux sinon. (Le noeud plus grand champion est un noeud élu champion qui servira à synchroniser le balayage dans certains cas. Entre deux numéros reçus par la source ou par un chef, son numéro est le plus grand.)

(i) Noeud(ronde commune) == le nombre de rondes à patienter avant de faire une action. À chaque ronde, les noeuds doivent décrémenter (ronde commune) afin de connaître la ronde pour débuter le balayage.

4. Message(destination) == source : signifie que le message est destiné à la source.
5. Message(numéro) == X : signifie que X est le nombre envoyé avec le message.

4.5 Description de l'algorithme

Les deux premières procédures de l'algorithme permettent d'élire les axes. Soit les axes internes lorsque la source est de degré 4, soit l'axe externe si la source est de degré 3 et finalement les deux axes externes si la source est de degré 2.

4.5.1 La procédure élection des axes internes

Cette procédure est utilisée lorsque la source est à l'intérieur de la grille (la source est un noeud de degré 4). L'élection permet de différencier l'axe vertical et l'axe horizontal, ce qui nous donnera quatre demi-axes. La procédure permet de transmettre différents messages et réponses aux noeuds, ce qui nous permettra de choisir avec certitude tous les noeuds situés sur les axes. En étudiant les grilles, nous avons découvert que certaines situations ne se produisent que sur les axes. Par exemple, les premiers noeuds informés par la source sont nécessairement sur les axes car ce sont les voisins de la source. Par la suite, nous nous servons de cette information pour sélectionner les voisins des voisins de la source. Nous avons donc utilisé ces situations pour sélectionner ces différents noeuds. Dans le futur, certains de ces noeuds pourront être appelés à faire la diffusion par balayage. (voir la figure 4.12)

1. La source envoie à tous ses voisins :
   (a) le message initial
   (b) compteur = 1
2. Un noeud v reçoit le message :
   (a) v(numéro) = compteur
   (b) compteur ++
   (c) envoyer le message initial avec le compteur à tous ses voisins
3. Les règles à suivre pour l'élection :
   (a) Si le noeud v est de degré 4 :
      i. Si v(numéro) est impair et que v reçoit à nouveau un message durant la ronde (v(numéro) + 2), il est donc un des noeuds élus (v(élu) = vrai). Il envoie un second message à la ronde (v(numéro) + 4) pour dire qu'il est un élu à tous ses voisins.
      (Il faut attendre une ronde (v(numéro) + 3) pour éviter les collisions.)

ii. Si v(numéro) est pair :

A. Si le noeud v reçoit un second message d'un noeud u élu (u(élu) == vrai) durant la ronde (v(numéro) + 3) et que v(numéro) == u(numéro) + 1, il est donc élu (v(élu) = vrai).

(b) Si le noeud v est de degré 3 :

i. Si v(numéro) est impair : il a reçu le message d'un noeud durant la ronde X. À la ronde suivante (X + 1), il envoie le message à tous ses voisins. Si durant la ronde X + 2, il ne reçoit rien, il est le chef de la ligne (v(chef ligne) = vrai).

ii. Si v(numéro) est pair : v a reçu un message durant la ronde X et que durant la ronde X + 3, il reçoit un message d'un noeud u élu tel que (u(numéro) == X - 1), alors il est le chef de la ligne (v(chef ligne) = vrai).

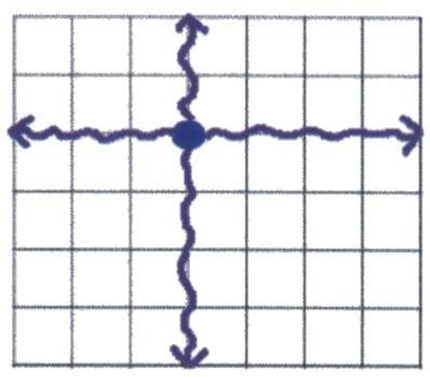

FIG. 4.12 – La procédure élection des axes internes

4.5.2 La procédure élection des axes externes

Cette procédure est utilisée quand la source est sur un côté de la grille (la source est un noeud de degré 3). L'élection permet de différencier l'axe sur lequel est situé la source par rapport aux trois autres côtés de la grille, c'est cet axe qui sera élu. Elle est aussi utilisée lorsque la source est dans un coin (noeud de degré 2). Dans ce cas, les deux axes contenant la source sont élus. La procédure permet de transmettre différents messages et réponses aux noeuds, ce qui nous permettra d'élire avec certitude tous les noeuds qui sont situés sur le(s) même(s) axe(s) que la source. Dans le futur, les noeuds de l'axe élu pourront être appelés à faire la diffusion par balayage. (voir la figure 4.13 et 4.14)

1. Le noeud u (u(source) == vrai ou u(plus petit champion) == vrai) envoie à tous ses voisins :
   (a) le message initial
   (b) compteur = 1
2. Les noeuds de degré 3 sont élus (v) :
   (a) v(numéro) = compteur
   (b) v(élu) = vrai
   (c) compteur ++
   (d) envoyer le message à N(v)
3. Les noeuds de degré 2 (u(degré) == 2) sont les noeuds chef de la ligne. (u(chef ligne) = vrai).
4. Lorsqu'un noeud reçoit le message pour une deuxième fois, il ne fait rien.
5. Les noeuds de degré 4 ne font rien.

Maintenant, le message a été transmis par la source et il y a eu élection des noeuds sur les axes. Par la suite, il doit y avoir un retour de message à la source et une élection du chef champion (sauf pour les cas de demi-axes égaux). Les deux procédures qui suivent permettent de retourner le message à la source et de sélectionner un chef champion.

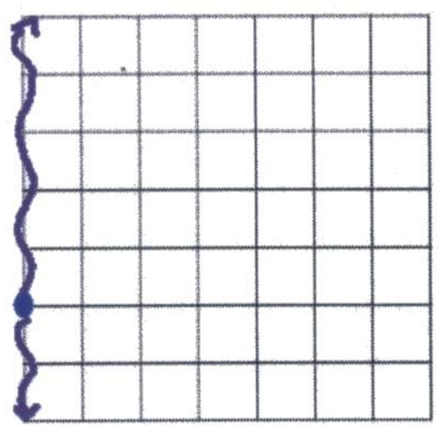

FIG. 4.13 – La procédure élection des axes externes

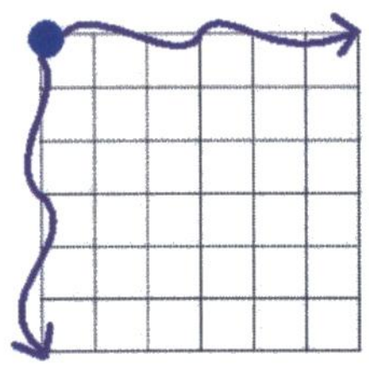

FIG. 4.14 – La procédure élection des axes externes

4.5.3 La procédure retour du message à la source

Cette procédure est utilisée lorsqu'un message est rendu à l'extrémité de la grille et que nous voulons en informer la source. Cette procédure permet d'envoyer un message vers la source afin que la source soit informée que cette étape de l'algorithme est atteinte. Avec ce retour de messages, nous profitons aussi du fait que nous pouvons obtenir certaines informations, par exemple, la distance entre un noeud externe et la source. (voir la figure 4.15)

1. Le noeud u chef de ligne envoie un message à tous ses voisins qui contient :
   (a) le nombre X où X == u(numéro)
   (b) destination(source)
2. Lorsqu'un noeud v élu (où v(élu) == vrai) reçoit le message, il le transmet à tous ses voisins.
3. Lorsqu'un noeud u non élu (où u(élu) == faux) reçoit le message, il ne fait rien
4. Le noeud v(source) == vrai reçoit un premier message contenant le numéro du premier chef.(v(numéro X) = message1(numéro))

5. Le noeud v(source) == vrai reçoit un seconde message contenant le numéro du second chef.(v (numéro Y) = message2(numéro))

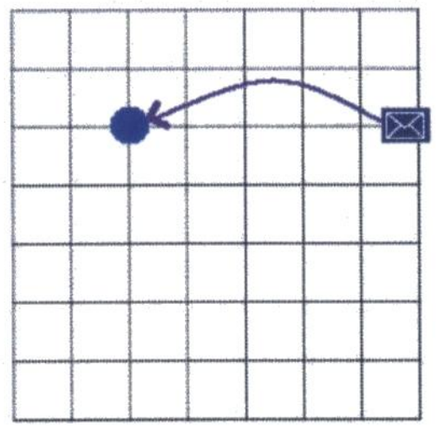

FIG. 4.15 – La procédure retour du message à la source

4.5.4 La procédure élection du plus grand ou plus petit chef champion

Cette procédure est utilisée lorsqu'un noeud reçoit deux messages de deux noeuds différents. Il regarde le numéro du message (c'est-à-dire le nombre de rondes qui sont passées entre la transmission de la source et la réception du noeud) et choisit celui avec le plus grand ou le plus petit numéro. Dans l'algorithme maître il y aura deux paramètres : plus grand ou plus petit ainsi que degré 2 ou 3. Cette procédure est utilisée lorsque nous voulons, par exemple, départager deux côtés de la grille. Le coin reçoit un premier message, quelques rondes plus tard, il en reçoit un deuxième. Par la suite, il prend une décision et envoie un message en disant que le noeud qui est à X distance de lui, est le noeud élu. (voir les figures 4.16 et 4.17) Les noeuds en vert vont élire les noeuds les plus petits (2) ou les noeuds les plus grands (6). D'habitude la source permet de départager les noeuds, mais dans ce cas précis, la source ne peut départager les noeuds. Lorsque les deux noeuds (2) vont communiquer avec elle, il y aura collision et la même chose arrivera lorsque les deux noeuds (6) essaieront de communiquer avec la source.

1. Le noeud u choisit le plus grand ou le plus petit numéro comme chef champion. (u(numéro X) u(numéro Y)). Ils ne peuvent pas être égaux car u n'aurait rien entendu.
2. Le noeud u envoie un message à tous ses voisins disant que le noeud de degré 2 ou 3 qui a le numéro min(X,Y) ou max(X,Y) est le chef champion (grand ou petit).
3. Lorsqu'un noeud v élu (où v(élu) == vrai) reçoit le message, il le transmet à tous ses voisins.
4. Lorsqu'un noeud u chef de ligne (u(chef ligne) == vrai) et que u(degré) == 2 ou 3 et que u(numéro) == min(X,Y) ou max(X,Y) alors u(plus grand champion) ou u(plus petit champion)= vrai.

Maintenant, il ne reste qu'à synchroniser tous les noeuds élus pour les avertir du numéro de la ronde de balayage et d'effectuer le balayage. La première procédure permet de synchroniser les noeuds élus et les deux suivantes permettent d'effectuer la diffusion par balayage.

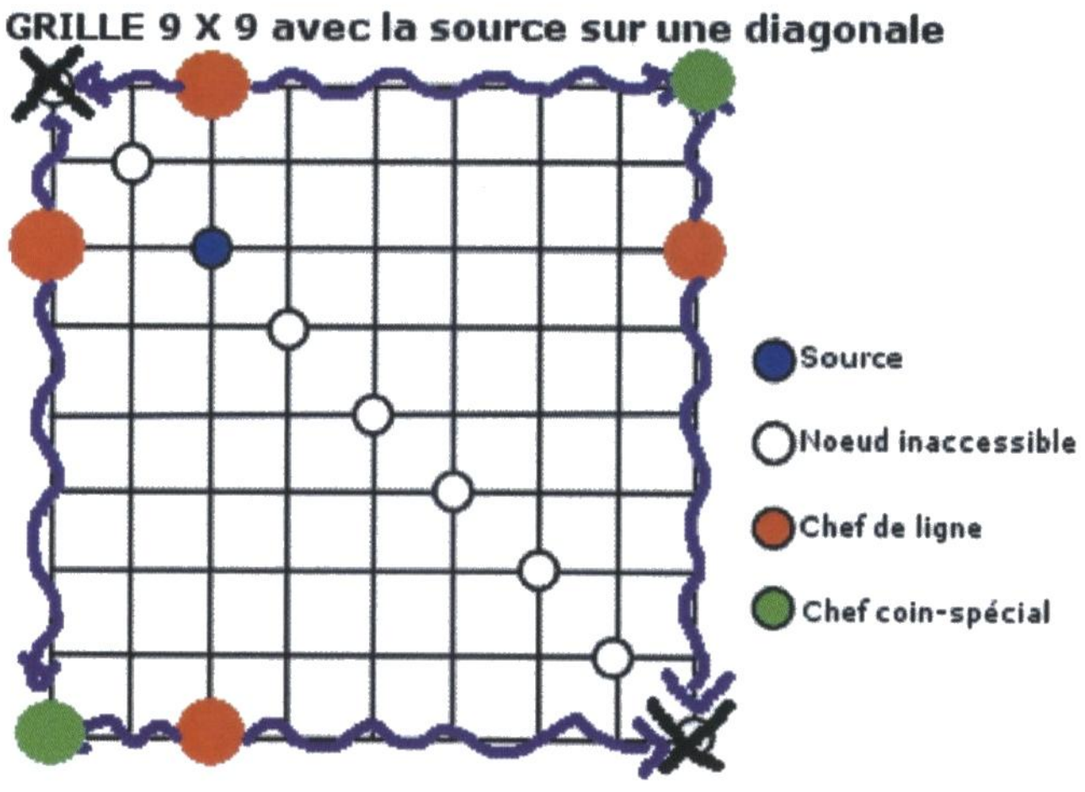

FIG. 4.16 – Élection du plus grand ou plus petit chef champion

4.5.5 La procédure synchronisation des chefs de lignes

Cette procédure est utilisée lorsque nous voulons qu'il y ait une synchronisation entre les chefs de lignes ainsi que les noeuds élus. Lorsque cette procédure est exécutée, les chefs de ligne (ou les chefs champions) transmettent un message à tous les noeuds élus afin de leur faire connaître le numéro de la ronde commune. Les noeuds vont conserver ce numéro de ronde et devront effectuer un balayage lorsque l'exécution de l'algorithme sera parvenu à cette ronde particulière. (voir la figure 4.18)

1. Le noeud u chef envoie à tous ses voisins :
   (a) un message de synchronisation
2. Un noeud v (où v(élu) == vrai) reçoit le message durant la ronde X :
   (a) v(ronde commune) = v(numéro) + X
   (b) envoyer le message à tous ses voisins
3. Lorsqu'un noeud u (où u(élu) == faux) ou u(chef ligne) == vrai reçoit le message :

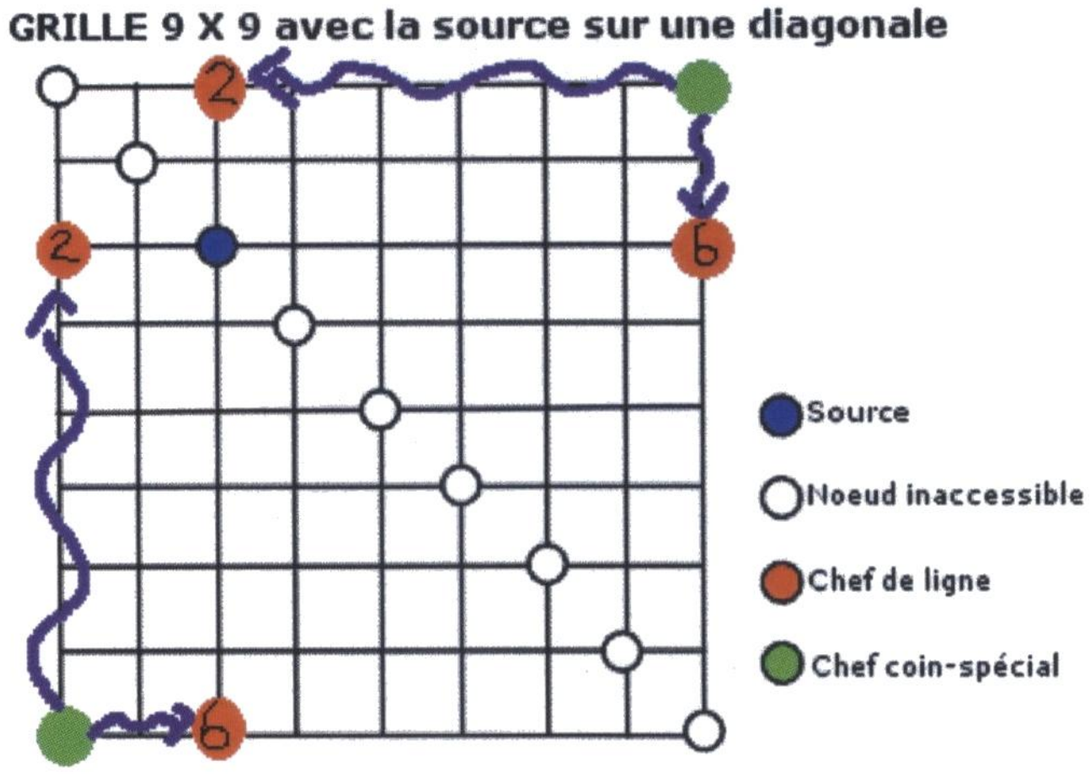

FIG. 4.17 – Élection du plus grand ou plus petit chef champion

(a) il ne fait rien

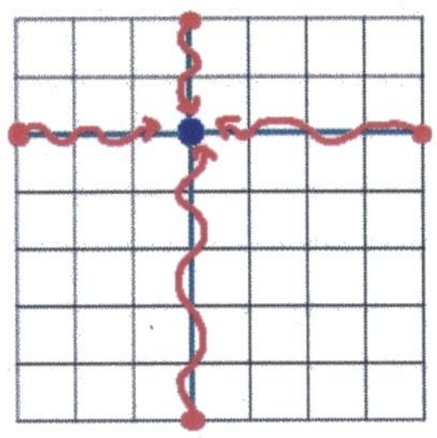

FIG. 4.18 – La procédure synchronisation des chefs de ligne

4.5.6 La procédure diffusion par balayage dans l'axe interne

Cette procédure est utilisée par la source lorsqu'elle est située sur un côté externe de la grille (la source est un noeud de degré 3). Elle permet d'effectuer la diffusion complète dans l'axe interne. C'est-à-dire, lorsque la source est située sur un axe externe de la grille, il y a tout de même un axe complètement interne (donc des noeuds de degré 4) dont la source fait partie. Cette procédure permet de transmettre le message à tous les noeuds de cet axe. (voir la figure 4.19)

1. La source envoie à tous ses voisins :
   (a) le message initial
2. Un noeud v reçoit le message :
   (a) s'il est de degré 4 (v(degré) == 4), il doit envoyer le message à tous ses voisins.
   (b) sinon, ne rien faire.
3. Si un noeud reçoit une deuxième fois le message, il ne fait rien de nouveau.

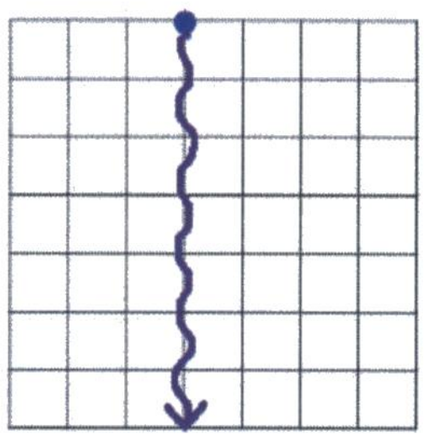

FIG. 4.19 – La procédure diffusion dans l'axe interne

4.5.7 La procédure diffusion du message par balayage

Cette procédure permet d'effectuer la diffusion lorsque la ronde commune est arrivée. Cette procédure fonctionne tout le temps à chaque début de rondes. Elle vérifie à chaque ronde si la ronde commune est arrivée, si ce n'est pas le cas, elle décrémente le nombre de rondes à attendre à chaque noeud élu. Lorsque la ronde commune est arrivée, tous les noeuds élus qui ont reçus un message de synchronisation vont effectuer la diffusion en même temps que leur chef. (voir la figure 4.20)

1. Lorsque v(ronde commune) == 0, tous les noeuds élus v (où v(élu) == vrai) envoient le message à tous leurs voisins
2. Lorsqu'un noeud non élu u (où u(élu) == faux) reçoit le message, il le transmet à tous ses voisins
3. Lorsqu'un noeud u élu (où v(élu) == vrai) reçoit le message, il ne fait rien.

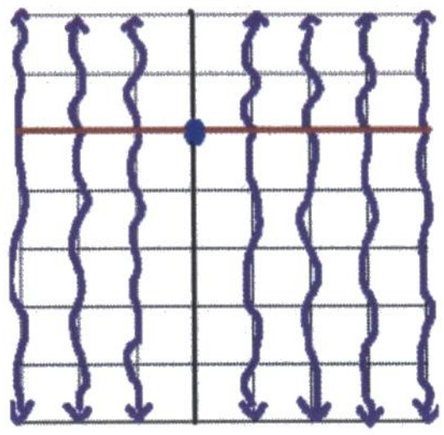

FIG. 4.20 – La procédure diffusion du message par balayage

Dans la majorité des situations, les procédures précédentes permettent d'effectuer la diffusion. Il ne reste qu'une situation dans laquelle nous avons besoin de faire deux autres élections afin de compléter la diffusion. Cette situation est celle où la grille est carrée et impaire et la source est située sur une des deux diagonales. (voir la figure 4.21).

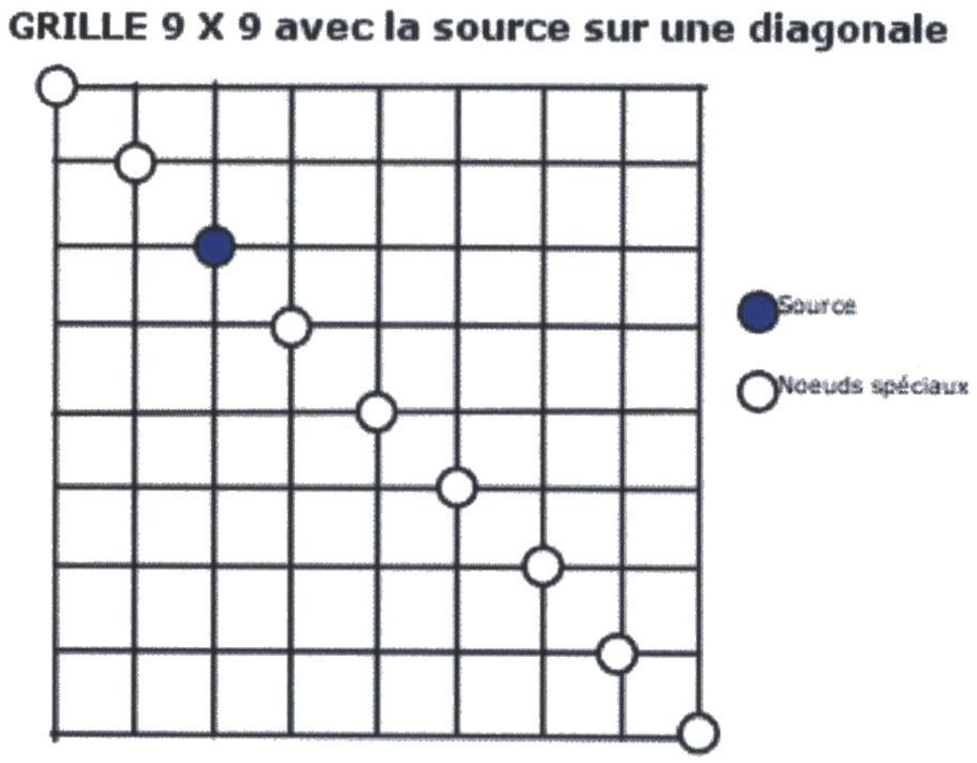

FIG. 4.21 – Grille carrée impaire avec la source située sur une diagonale

Premièrement, il y a eu une élection interne, les noeuds en rouge sont les chefs de lignes (voir la figure 4.22). Par contre, il est impossible de se servir d'un retour à la source car les demis-axes sont égaux deux à deux, il y aura donc toujours une collision lorsque le message reviendra vers la source.

Cette difficulté est surmontée en se servant des noeuds chefs de ligne (en rouge). Les chefs de ligne vont démarrer une autre élection, celle des chefs-coins spéciaux. Les deux chefs-coins spéciaux (en vert) ont bien reçu le message, alors que les deux autres coins n'ont rien entendu.(voir la figure 4.23).

Par la suite, les deux chefs-coins spéciaux vont effectuer l'élection du plus petit chef champion. Ils vont transmettre un message qui indique que le noeud chef de ligne avec le numéro le plus petit (dans ce cas-ci c'est le numéro 2), est le chef-champion petit. (voir la figure 4.24).

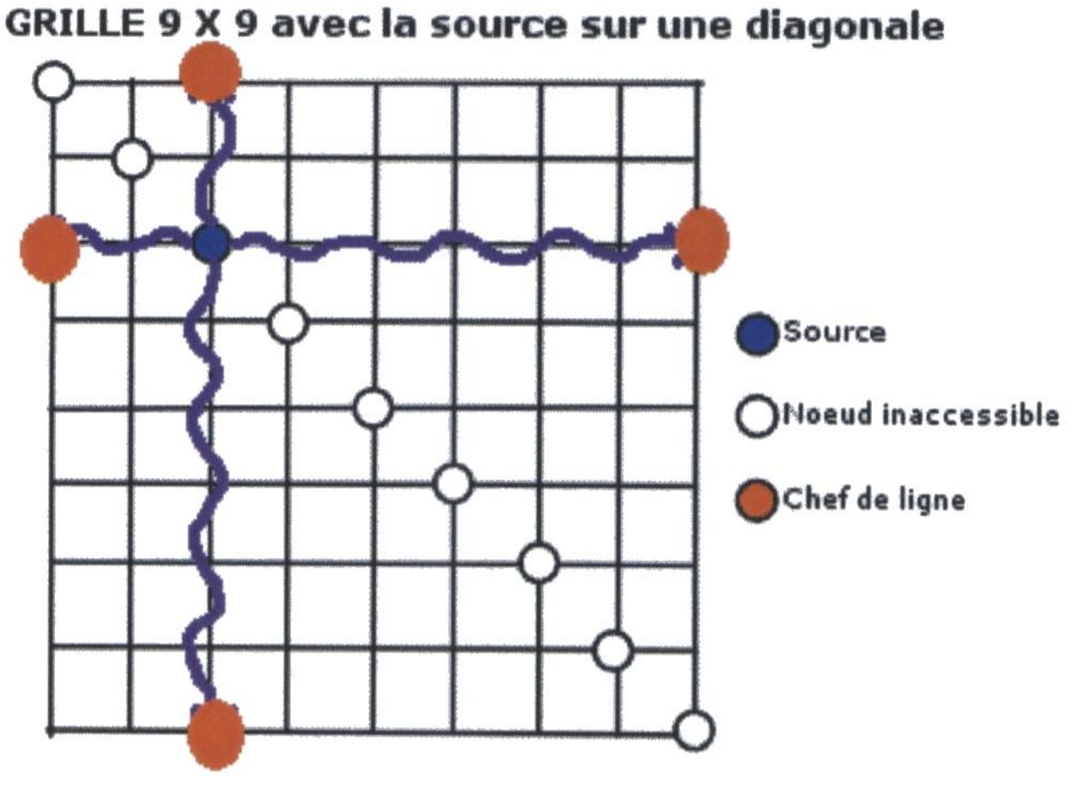

FIG. 4.22 – Élection dans une grille carrée impaire avec la source située sur une diagonale

Ensuite, le chef-champion petit (en rouge) synchronise tous les noeuds élus de son axe externe (voir la figure 4.25).

Ensuite, tous les noeuds synchronisés font la diffusion par balayage (voir la figure 4.26).

4.5.8 La procédure élection des chefs-coins spéciaux

Cette procédure est utilisée pour élire deux coins spéciaux. Ces coins nous permettront de différencier les deux côtés et plus tard de pouvoir choisir les noeuds champions. Nous utilisons cette procédure lorsque les coins ne sont pas tous situés à égale distance de la source. Il est alors possible de les différencier et d'élire les chefs. Grâce à ces chefs, il sera ensuite possible de sélectionner certains noeuds qui seront responsables de démarrer la diffusion par balayage. (voir la figure 4.23)

1. La source envoie à tous ses voisins :
   (a) le message initial
   (b) compteur = 1

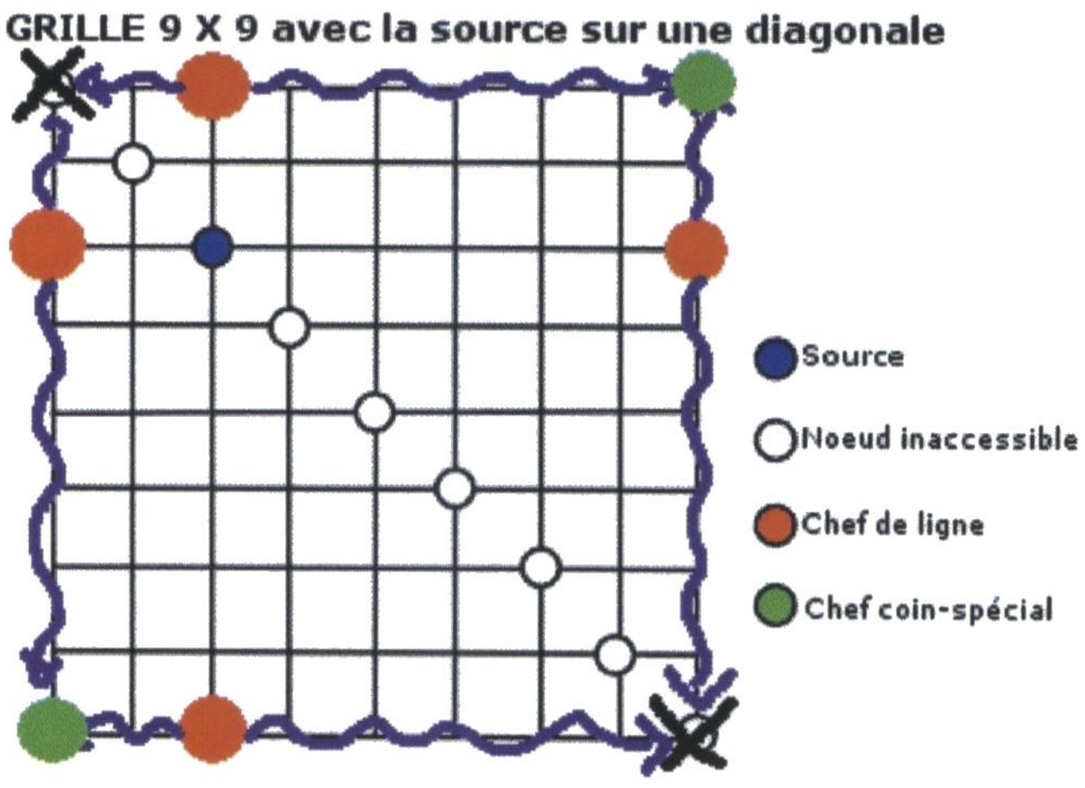

FIG. 4.23 – Élection dans une grille carrée impaire avec la source située sur une diagonale

2. Les noeuds de degré 3 sont élus (v) :
   (a) v(numéro) = compteur
   (b) v(élu) = vrai
   (c) compteur ++
   (d) envoyer le message à tous ses voisins
3. Les noeuds de degré 2 (u(degré) == 2) sont les noeuds chefs spéciaux de la grille. (u(chef spécial) = vrai :
   (a) Le noeud u chef spécial (u(chef spécial) == vrai) reçoit un premier message contenant le numéro du premier chef.(u(numéro X) = message1(numéro))
   (b) Le noeud u chef spécial (u(chef spécial) == vrai reçoit un second message contenant le numéro du second chef.(u (numéro Y) = message2(numéro))
4. Lorsqu'un noeud reçoit le message pour une deuxième fois, il ne fait rien.
5. Les noeuds de degré 4 ne font rien.

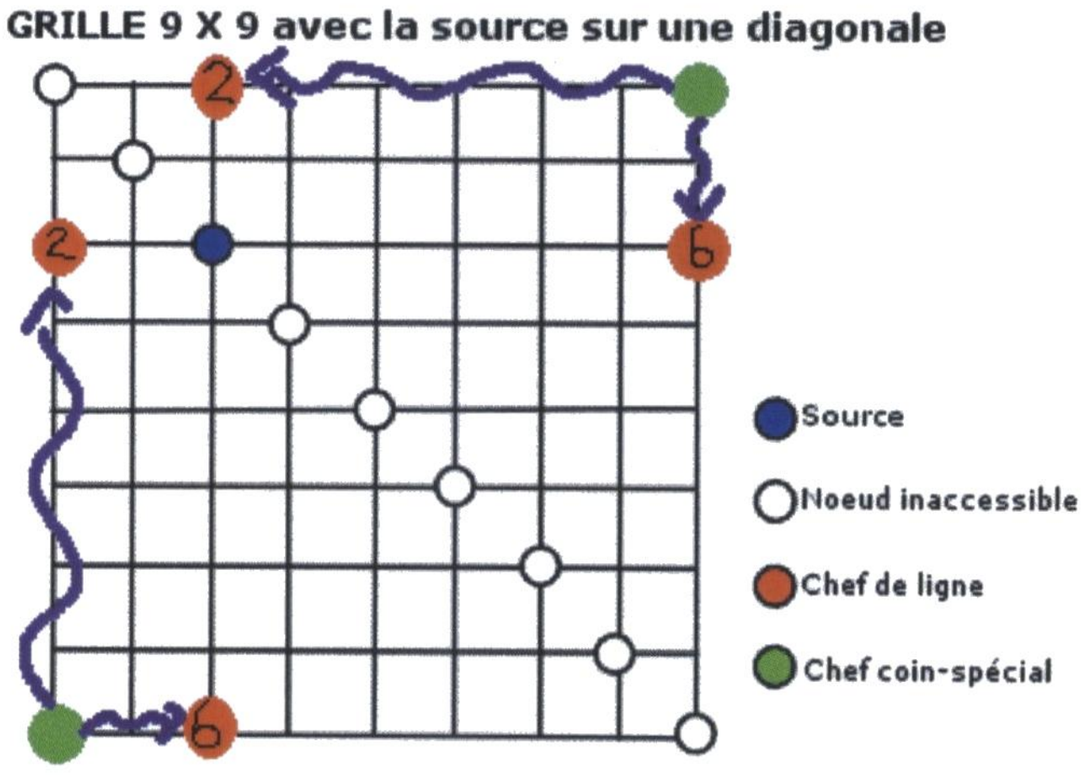

FIG. 4.24 – Élection dans une grille carrée impaire avec la source située sur une diagonale

4.5.9 La procédure multiplexage

Cette procédure est utilisée lorsque nous avons plusieurs procédures que nous voulons exécuter en parallèle. Elle permet, par exemple, d'exécuter une procédure aux rondes paires et une autre aux rondes impaires. Ce qui nous permet d'appliquer plusieurs solutions pour résoudre notre problème car nous ne savons pas toujours dans quelle situation nous sommes. Nous savons que peu importe la situation, nous sommes toujours dans un des huit cas possibles. Grâce au multiplexage, nous parvenons à faires les procédures pour les huit cas et nous pouvons toujours être certain d'avoir atteint le maximum de noeuds possibles. Par contre, lorsque nous savons exactement dans quelle situation nous nous situons, il est préférable de simplement utiliser les procédures nécessaires. Dans ce cas, le multiplexage n'est pas nécessaire.

1. Fusion (procédure-1, procédure-2)
2. Pour chaque i :
   (a) Pendant la ronde 2 i - 1, exécuter la ronde i de la procédure-1.

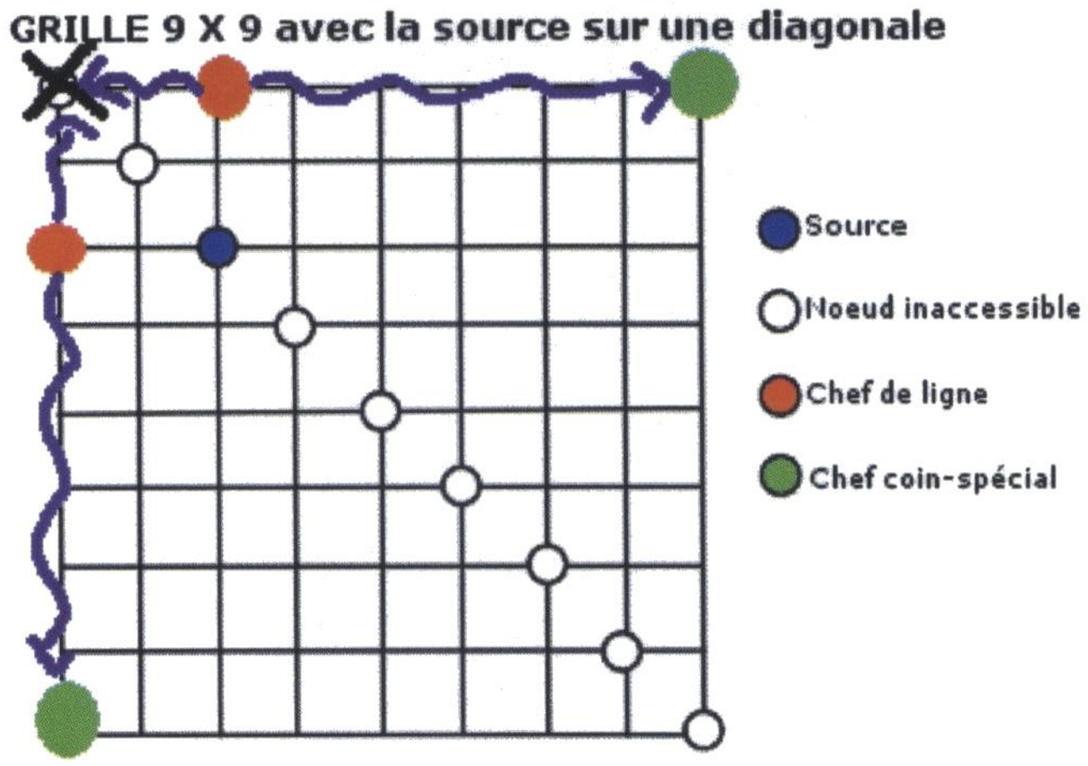

FIG. 4.25 – Synchronisation dans une grille carrée impaire avec la source située sur une diagonale

(b) Pendant la ronde 2 i, exécuter la ronde i de la procédure-2.

1. Multiplex (procédure-1, procédure-2, ..., procédure-N)
2. Lorsque N = 2 :
   (a) Fusion (procédure-1, procédure-2)
3. sinon :
   (a) Fusion (Multiplex(procédure-1, ..., procédure-N-1), procédure-N)

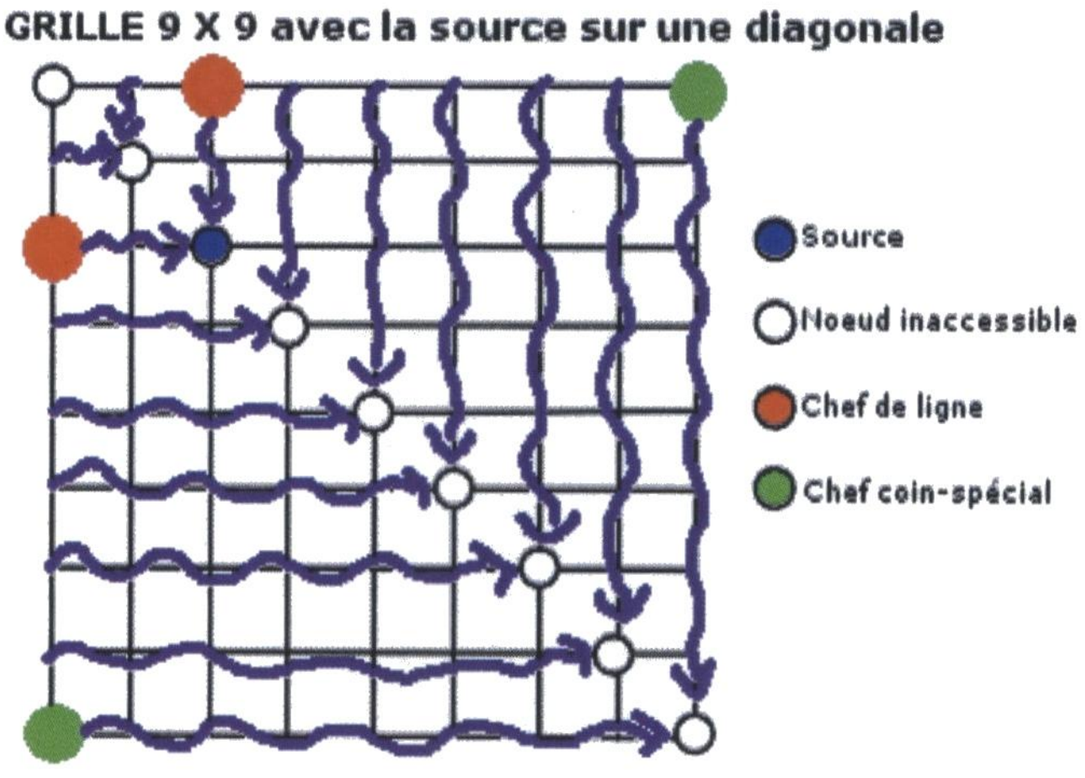

FIG. 4.26 – Balayage dans une grille carrée impaire avec la source située sur une diagonale

4.5.10 Concaténation des procédures

Les procédures qui doivent être exécutées une à la suite de l'autre sont maintenant regroupées en une seule fonction. Ce sont ces fonctions qui seront multiplexées à d'autres fonctions. Nous avons regroupé les procédures qui représentent un seul des huit cas de la grille rectangulaire. Ceci est pour faciliter la lecture et la compréhension de l'algorithme général de diffusion.

1. Fonction 1 : cas où les deux axes sont de grandeurs différentes (source degré 2) :
   (a) procédure *retour du message à la source*
   (b) procédure *élection du chef champion (chef = petit, degré = 2)*
   (c) procédure *synchronisation du ou des chef(s) champion(s)*
   (d) procédure *diffusion du message par balayage*
2. Fonction 2 : cas où les deux axes sont égaux (source degré 2) :
   (a) procédure *synchronisation des chefs de lignes*

(b) procédure *diffusion du message par balayage*

3. Fonction 3 : cas où les deux demi-axes sont égaux(source degré 3) :

   (a) procédure *synchronisation des chefs de lignes*

   (b) procédure *diffusion du message par balayage*

4. Fonction 4 : diffusion de la ligne ou de la colonne de la source (suite du cas où les deux demi-axes sont égaux (source degré 3)) :

   (a) procédure *diffusion par balayage dans l'axe interne*

5. Fonction 5 : multiplexage (suite du cas où les deux demi-axes sont égaux (source degré 3)) :

   (a) procédure *multiplexage* (Fonction-3, Fonction-4, 2).

6. Fonction 6 : cas où il y a un demi-axe de longueur unique (source degré 3) :

   (a) procédure *retour du message à la source*

   (b) procédure *élection du chef champion (chef = petit, degré = 2)*

   (c) procédure *élection des axe externes* par le plus petit chef champion

   (d) procédure *synchronisation des chefs de lignes*

   (e) procédure *diffusion du message par balayage*

7. Fonction 7 : cas où il y a un demi-axe de longueur différente (source degré 4) :

   (a) procédure *retour du message à la source*

   (b) procédure *élection du chef champion (chef = petit, degré = 3)*

   (c) Le noeud champion deviendra la nouvelle source et nous pouvons maintenant nous référer au cas d'une source de degré 3 (u(source) = faux et v(plus petit champion) == vrai implique que v(source) = vrai).

   (d) procédure *élection des axes externes*

   (e) procédure *multiplexage* (Fonction-4, Fonction-6, 2).

8. Fonction 8 : cas où les 4 demi-axes sont égaux (source degré 4) :

   (a) procédure *synchronisation des chefs de lignes*

(b) procédure *diffusion du message par balayage*

9. Fonction 9 : Si les axes sont égaux deux à deux (opposés - égaux face-à-face) (source degré 4) :

   (a) procédure *élection des axes internes*

   (b) Lorsqu'un noeud de degré 3 est élu, il attend W rondes où W = (numéro) et il démarre (il devient la source de cette procédure) la procédure *élection des axes externes.*

   (c) procédure *synchronisation des chefs de lignes*

   (d) procédure *diffusion du message par balayage*

10. Fonction 10 : Si les axes sont égaux deux à deux (adjacents - égaux en coin) (source degré 4) :

    (a) procédure *élection des axes internes*

    (b) Lorsqu'un noeud de degré 3 est élu, il attend W rondes où W = (numéro) et il démarre (il devient la source de cette procédure) la procédure *élection des chefs-coins spéciaux.*

    (c) Fonction 11 : cas du plus petit champion (nous effectuons deux balayages complets afin de maximiser le nombre de noeuds informés)

        i. Les chefs-coins spéciaux effectuent la procédure *élection du chef champion (chef = petit, degré = 3)*

        ii. procédure *synchronisation du ou des chef(s) champion(s)* (par les axes externes)

        iii. procédure *diffusion du message par balayage*

4.5.11 L'algorithme maître : diffusion dans la grille rectangulaire

C'est le regroupement de toutes les procédures nécessaires pour la diffusion. Tous les cas possibles sont traités dans cet algorithme et en y suivant l'ordre nous pourrons effectuer la diffusion dans n'importe quelle grille rectangulaire pour chaque position de la source. La diffusion sera parfois complète ou parfois il restera certains noeuds qui ne seront pas informés. Cet algorithme est utilisé lorsque la position de la source est inconnue.

1. Si la source est de degré 2 (dans un coin) :
   (a) procédure *élection des axes externes*
   (b) procédure *multiplexage* (Fonction-1, Fonction-2, 2).
2. Si la source est de degré 3 (sur le bord de la grille) :
   (a) procédure *élection des axes externes*
   (b) procédure *multiplexage* (Fonction-3, Fonction-4, Fonction-6, 3).
3. Si la source est de degré 4 (à l'intérieur de la grille)
   (a) procédure *élection des axes internes*
   (b) procédure *multiplexage* (Fonction-7, Fonction-8, Fonction-9, Fonction-10, 4)
   (c) procédure *multiplexage* (Fonction-11, Fonction-12, 2).

4.6 Les résultats

Voici les résultats lorsque nous utilisons cet algorithme dans un réseau radio de type grille rectangulaire.

1. La source est dans un coin de la grille (degré 2) :
   (a) Les deux axes sont de longueurs différentes : tous les noeuds sont atteints
   (b) Les deux axes sont de même longueur (grille carrée) : les noeuds de la diagonale (allant de la source au coin opposé) ne sont pas atteints. Ce sont les noeuds inaccessibles. Tous les autres noeuds sont atteints.
2. La source est sur un côté de la grille (degré 3) :
   (a) Tous les noeuds sont atteints
3. La source est de degré 4 :
   (a) Les 4 demi-axes sont égaux (grille carrée) : les noeuds sur les deux diagonales ne sont pas atteints. Ce sont les noeuds inaccessibles. Tous les autres noeuds sont atteints.
   (b) Les demi-axes sont égaux 2 à 2
      i. demi-axes égaux (opposés - égaux face-à-face) : tous les noeuds sont atteints
      ii. demi-axes égaux (adjacents - égaux en coin) (grille carrée) : Les noeuds de la diagonale contenant la source ne sont pas atteints. Ce sont les noeuds inaccessibles. Tous les autres noeuds sont atteints.
   (c) Il y a au moins un demi-axe qui est de longueur unique : tous les noeuds sont atteints

Les seuls cas où il y a des noeuds **inaccessibles** sont ceux où la grille est carrée et la source est sur une ou deux diagonales. Alors cette (ces) diagonale(s) forment l'ensemble des noeuds inaccessibles (à l'exception de la source elle-même).

Il est aussi important de se souvenir que notre algorithme possède deux volets. Le premier est utilisé lorsque nous avons toutes les informations nécessaires sur le réseau. Comme nous savons exactement dans quelle situation nous sommes, nous n'avons pas à multiplexer toutes les fonctions, ce qui fait que la diffusion peut se faire beaucoup plus rapidement.

Pour l'autre volet (présenté comme l'algorithme maître), nous n'avons pas toutes les informations nécessaires sur le réseau donc nous devons multiplexer plusieurs fonctions de l'algorithme pour être certain d'avoir atteint tous les noeuds accessibles.

4.7 Conclusion

Dans ce chapitre, nous avons développé un algorithme de diffusion de messages pour les réseaux radio anonymes de type grille rectangulaire. Notre objectif était de trouver un algorithme qui permettait d'informer tous les noeuds accessibles dans une grille rectangulaire. Afin d'atteindre notre objectif nous avons divisé le problème en plusieurs sous-problèmes.

Chaque sous-problème représente une situation particulière de la grille. Afin de trouver un algorithme complet, nous avons commencé à chercher une solution pour chaque situation. Pour chacune des solution le grand principle reste le même. Il faut sélectionner certains noeuds qui ont une position particulière et par la suite, nous pouvons nous servir de ces noeuds particuliers pour faire une diffusion par balayage. Par la suite, nous avons mis toutes ces solutions ensembles, en applicant le multiplexage ce qui nous a permis de résoudre notre problème sans savoir dans quelle situation nous sommes.

Notre algorithme permet d'informer tous les noeuds accessibles de n'importe quel réseau radio grille rectangulaire. Dans le prochain chapitre, nous allons discuter des simulations de notre algorithme de diffusion qui permettent de valider son exactitude et d'estimer son temps de fonctionnement en pratique.

Chapitre 5

Simulations de l'algorithme de diffusion

5.1 Introduction

Afin d'illustrer le fonctionnement de notre algorithme décrit au chapitre précédent, nous avons développé une application en Java qui l'implante. Avec cette application, nous avons fait plusieurs simulations de diffusion dans différentes grilles afin d'évaluer l'efficacité de l'algorithme à l'aide d'une technique expérimentale. Selon les paramètres que nous fournissons à chaque exécution, cette application effectue la diffusion et nous indique le nombre de rondes nécessaires pour informer tous les noeuds accessibles. Dans le cas des réseaux radio, il est plus intéressant d'analyser le nombre de rondes que le nombre de messages. Lorsqu'un noeud transmet un message, tous ses voisins peuvent l'entendre s'il n'y a pas d'interférence. Que le message ait été reçu par dix récepteurs ou par un seul, ce n'est pas plus coûteux. Cela n'a donc pas d'influence sur l'efficacité de l'algorithme. Alors que le nombre de rondes nécessaires pour effectuer la diffusion est une mesure plus naturelle de l'efficacité de l'algorithme. Nous savons que s'il n'y avait pas d'interférence, la diffusion devrait prendre R *rondes* où R est la distance entre la source et le noeud le plus éloigné (le rayon de la grille). Donc nous comparerons le temps de la diffusion à R.

Ce chapitre est divisé comme suit : La section 5.2 explique l'application que nous avons créée pour simuler notre algorithme de diffusion. Puis la section 5.3 expose les résultats des simulations.

5.2 Description de l'application

Notre application permet, premièrement, de valider et d'analyser le fonctionnement de notre algorithme de diffusion et deuxièmement d'illustrer les diffusions dans les réseaux radio concrets de type grille rectangulaire. Étant donné le grand nombre de transmissions de messages nécessaires pour effectuer la simulation de la diffusion dans les réseaux, nous avons dû simplifier l'affichage des transmissions le plus que possible. Sinon, il devenait impossible de suivre le fonctionnement de l'algorithme. Voici les grandes étapes d'utilisation de notre application.

Premièrement pour définir la grille et la source, nous pouvons générer aléatoirement un réseau ainsi que la position de la source. Lors des nombreuses simulations, nous avons utilisé majoritairement cette technique. Nous pouvons aussi, fournir en paramètre le nombre de colonnes, de lignes ainsi que la position de la source.

Deuxièmement, nous avons dû prendre certaines décisions quant à l'affichage graphique. Nous devons déterminer si nous voulons avoir un affichage graphique ou non. L'affichage graphique fonctionne pour des grilles de 15 par 15 au maximum. Lorsque nous voulons simuler de plus grandes grilles, il sera nécessaire de désactiver l'affichage graphique car il deviendrait illisible.

Troisièmement, nous devons choisir si la position de la source est connue ou inconnue. Lorsque la position est connue, seules les procédures nécessaires de l'algorithme seront exécutées. Alors que si la position de la source est inconnue, l'algorithme entier sera exécuté, en effectuant le multiplexage de toutes les procédures.

Lorsque nous démarrons la diffusion, l'application affiche à l'écran le réseau et débute la diffusion. Lorsqu'un noeud est informé, il change de couleur. Les noeuds particuliers (la source, les chefs et les élus) sont affichés d'une couleur particulière. Lorsqu'il y a une collision, il y a un symbole de collision qui clignote, ce qui permet de rapidement voir toutes les collisions durant une ronde précise. L'affichage permet de voir une ronde, ensuite il y a un délai de quelques secondes avant d'afficher la prochaine ronde. Une situation particulière survient lorsque nous décidons d'effectuer une diffusion avec une source inconnue. Afin de bien illustrer les rondes de multiplexage, nous avons changé la couleur du réseau à chaque fois qu'il y a un multiplexage.

5.3 Les simulations

Afin d'estimer l'efficacité de notre algorithme, nous avons fait plusieurs simulations. Nous avons estimé l'efficacité lorsque la position de la source était connue (voir le tableau 5.1). Voici notre démarche : Nous avons généré une centaine de grilles pour chacun des huit cas possibles (source de degré 2 avec axes égaux ou inégaux, source de degré 3 avec les demi-axes égaux ou inégaux, source de degré 4 avec les quatre demi-axes égaux ou quatre demi-axes égaux deux à deux en coin ou face-à-face et au moins un demi-axe de longueur unique). Afin de déterminer l'efficacité de l'algorithme dans chaque grille, nous avons divisé le nombre de rondes utilisés par le rayon de la grille (T / R). En absence de collisions, la diffusion utiliserait exactement R rondes et R est la bonde inférieure sur le temps de la diffusion. Par contre, dans notre situation, ce résultat est impossible avec les collisions (interférences) dans les réseaux radio. Avec cette division, nous obtenons toujours un résultat supérieur à un, ce résultat que nous appelons surplus, nous permet d'analyser l'efficacité de cet algorithme.

Après ces simulations de quelques centaines de grilles, nous avons compilé les résultats et nous présentons leur moyenne dans le tableau 5.1. Les résultats y sont présentés individuellement pour tous les différents types de grilles car il y a des écarts considérables entre chacun des cas. La situation la plus efficace est celle où les deux axes sont égaux. La diffusion s'effectue en moins de *2R rondes*. Lorsque les deux axes sont égaux et que la source est de degré deux ou trois, il est facile de sélectionner tous les noeuds des deux axes et un seul balayage sera nécessaire. Les diffusions les plus longues sont celles où la source est de degré quatre. Il faut utiliser beaucoup plus de rondes car il est plus compliqué de choisir certains noeuds lorsqu'il y a quatre demi-axes. Nous ne voulons pas que tous les noeuds qui sont sur la même ligne et la même colonne que la source soient élus. Nous utilisons plusieurs détours pour y arriver, ce qui prend plusieurs rondes supplémentaires.

Évidemment, les diffusions lorsque la position de la source est connue sont beaucoup plus rapides car seules les procédures nécessaires sont exécutées. Lorsque la position de la source est inconnue, mais que son degré est connu, voici les différents résultats (voir le tableau 5.2). Plusieurs rondes supplémentaires sont nécessaires comme nous ne savons pas dans quelle situation nous sommes, nous devons parcourir l'algorithme au complet pour s'assurer d'informer tous les noeuds accessibles. Par contre, comme le degré de la source est connue, nous pouvons laisser tomber toutes les parties de l'algorithme qui s'occupent des autres degrés.

Les situations	Nombre de rondes / rayon
Les deux axes sont égaux (degré 2)	1.80
Les deux axes sont différents (degré 2)	1.90
Les deux demi-axes sont différents (degré 3)	2.35
Les deux demi-axes sont égaux (degré 3)	1.95
Les 4 demi-axes sont égaux (degré 4)	3.05
Les demi-axes égaux (face-à-face) (degré 4)	3.4
Les demi-axes égaux (coin) (degré 4)	3.6
Un demi-axe est différent(degré 4)	3.2
La moyenne de tous les cas	**2.65**

TAB. 5.1 – Résultats des simulations lorsque la position de la source est connue

Les situations	Nombre de rondes / rayon
Source de degré 2	4.80
Source de degré 3	5.21
Source de degré 4	12
La moyenne	**7.33**

TAB. 5.2 – Résultats des simulations lorsque la position de la source est inconnu

Nous voulions aussi vérifier s'il y avait une corrélation entre le rayon de la grille et T/R (nombre de rondes divisé par le rayon de la grille). Nous avons généré aléatoirement cinquante grilles différentes de rayon 10, 15, 20 jusqu'à 60. Nous avons compilé les différents résultats (voir la figure 5.1). Nous avons pu constater qu'il n'y a pas vraiment de corrélation entre R et T/R. Peu importe le rayon de la grille, T/R reste près d'une constante. Elle varie entre 2.5 et 2.8, ce qui donne une moyenne de 2.65. En conclusion, que le rayon des grilles soit petit ou grand, notre algorithme parvient à effectuer la diffusion en plus ou moins 2.65 fois le rayon.

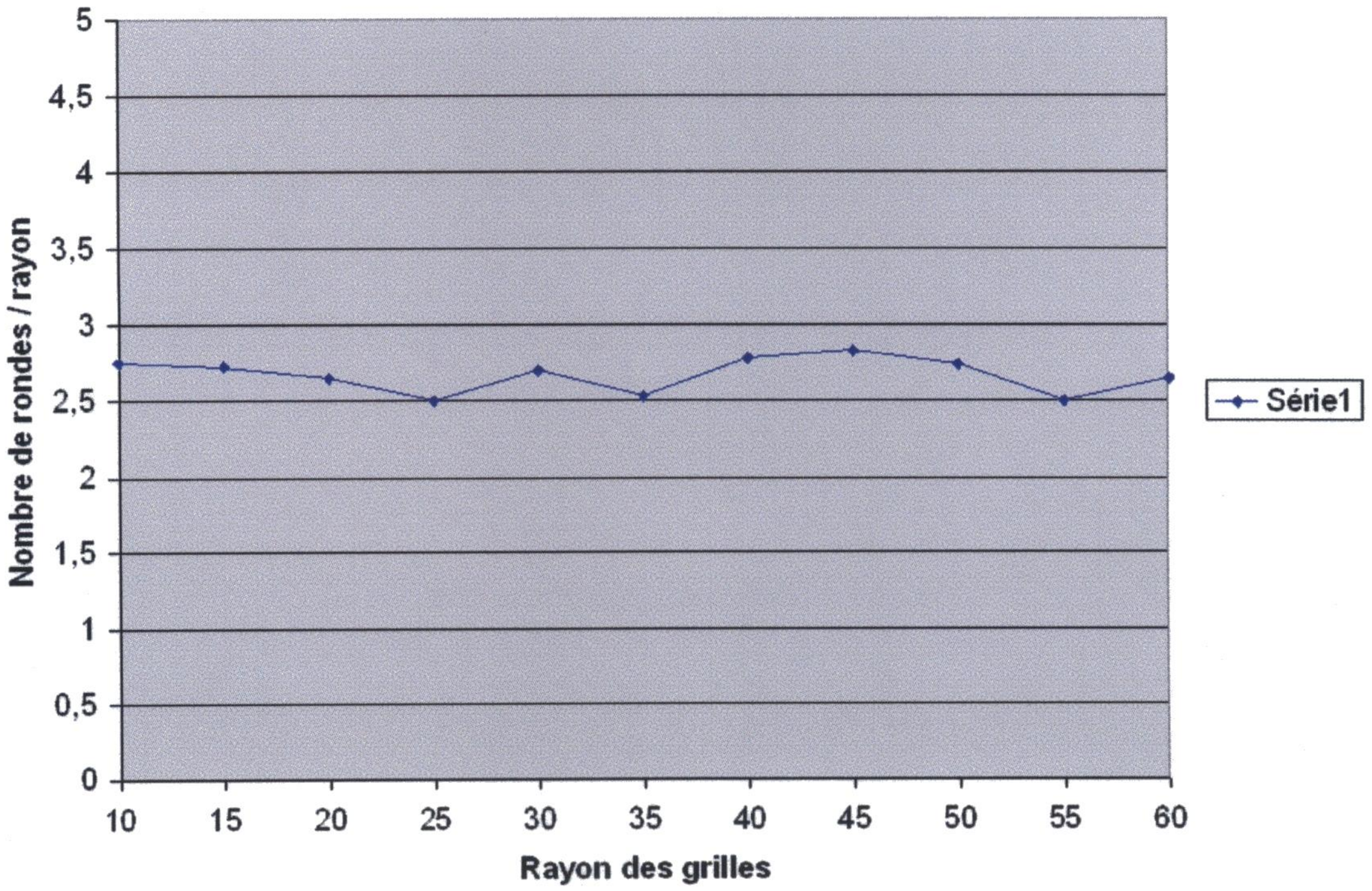

FIG. 5.1 – Relation entre le rayon des grilles et le nombre de rondes divisé par le rayon de la grille

Lors de nos simulations, nous avons aussi vérifié les noeuds qui n'étaient jamais informés. Les résultats sont exactement les mêmes que nous avions obtenus durant l'analyse du problème et la conception de l'algorithme. Les **noeuds inaccessibles** sont les noeuds qui font partis d'un ensemble de noeuds spéciaux. Ces ensembles sont créés lorsque la grille est carrée et que la source est située sur une ou deux diagonale(s). Dans la tableau 5.3, nous

pouvons voir que le nombre de noeuds inaccessibles est soit 0, soit *m-1*, soit *2(m-1)* où m est la dimension de la grille carrée.

Les situations	noeuds non informés
Les deux axes sont égaux (degré 2)	(m - 1)
Les deux axes sont différents (degré 2)	0
Les deux demi-axes sont différents (degré 3)	0
Les deux demi-axes sont égaux (degré 3)	0
Les 4 demi-axes sont égaux (degré 4)	2(m - 1)
Les demi-axes égaux (face-à-face) (degré 4)	0
Les demi-axes égaux (coin) (degré 4)	(m - 1)
Un demi-axe est différent(degré 4)	0

TAB. 5.3 – Nombre de noeuds non informés

5.4 Conclusion

Pendant l'étape des simulations, nous avons pu visualiser le fonctionnement de l'algorithme et s'assurer qu'il fonctionnait comme nous l'avions prévu. Nous avons aussi utilisé les simulations pour calculer expérimentalement l'efficacité de notre algorithme et par le fait même, comparer nos résultats avec le rayon de la grille. Nous avons aussi vérifié que les noeuds inaccessibles étaient exactement les mêmes que ceux que nous avions identifiés théoriquement. Notre objectif était d'atteindre tous les noeuds accessibles et les simulations montrent que ce but est accompli dans 100% des cas.

La complexité du temps de fonctionnement de notre diffusion a été expérimentalement confirmée comme étant proche de linéaire en grandeur du rayon R de la grille.

Chapitre 6

Conclusion

L'objectif de ce mémoire était d'étudier les réseaux radio anonymes de type grille rectangulaire et de fournir une solution pour effectuer une diffusion. Donc, le point central de ce mémoire était la construction d'un algorithme de diffusion qui informe tous les noeuds accessibles d'un réseau à topologie de grille rectangulaire arbitraire avec une position arbitraire de la source. Afin d'y arriver plusieurs étapes précédentes et subséquentes à l'algorithme ont été réalisées. Voici les grandes lignes de notre recherche.

Au chapitre 3, nous avons prouvé que certains noeuds étaient des noeuds inaccessibles, c'est-à-dire qu'ils seront toujours impossibles à atteindre, peu importe l'algorithme utilisé. Nous avons établi une condition suffisante pour qu'un noeud du réseau radio anonyme représenté par le graphe G soit impossible à informer par n'importe quel algorithme. Afin de prouver ce résultat, nous avons défini l'histoire d'un noeud comme étant la séquence de tous les messages qu'il a reçus précédemment. La seule façon pour un noeud de décider s'il doit transmettre ou non c'est de regarder son histoire (historique des communications) et de prendre une décision. Nous avons prouvé par induction que certains noeuds dits semblables auront toujours la même histoire et donc vont toujours transmettre ou se taire en même temps. Ceci rend certains autres noeuds inaccessibles. Ce résultat est valide pour tous les réseaux radio anonymes, mais nous l'utilisons seulement dans le cas des grilles. Il nous a permis de caractériser les noeuds inaccessibles dans chaque grille.

Au chapitre 4, nous avons présenté notre algorithme de diffusion pour les réseaux radio anonymes de type grille rectangulaire. Notre algorithme est constitué de beaucoup de procédures. Ces procédures servent à sélectionner ou informer les noeuds. En étudiant les grilles, nous avons trouvé qu'il y avait huit situations possibles. Nous voulions toujours parvenir

à informer tous les noeuds accessibles sans toutefois toujours savoir dans quelle situation nous étions. Suite à cette étude des grilles, nous avons trouvé un algorithme de diffusion pour chacune des situations. Toutes nos solutions suivent toujours la même idée : la source transmet un message, il y a élection de certains noeuds particuliers, les noeuds particuliers sont responsables de démarrer la diffusion par balayage tous en même temps. La difficulté de cette stratégie est : comment parvenir à sélectionner certains noeuds alors qu'ils sont tous anonymes et lesquels il faudra sélectionner. Lorsque la position de la source est inconnue, nous procédons au multiplexage des procédures qui servent à résoudre tous les cas possibles. Alors nous pouvons garantir que tous les noeuds accessibles seront toujours atteints car nous avons nécessairement effectué la procédure qui fait la diffusion de ce cas malgré qu'il soit inconnu (toutes les procédures ont été exécutées en multiplexage).

Au chapitre 5, nous avons discuté des simulations de notre algorithme. Nous avons fait une application en Java afin de simuler notre algorithme de diffusion avec différentes grilles. Nous avons fait plusieurs simulations et nous avons compilé les résultats afin d'estimer l'efficacité de notre algorithme. Notre objectif était d'atteindre tous les noeuds accessibles et les simulations montrent que ce but est accompli dans 100% des cas. La complexité du temps de fonctionnement de notre diffusion a été expérimentalement confirmée comme étant proche de linéaire en grandeur du rayon R de la grille.

Nos objectifs principaux étaient :

- Étudier la faisabilité de la diffusion dans un réseau radio de type grille rectangulaire. C'est-à-dire trouver les noeuds qui sont accessibles ainsi que ceux qui sont toujours inaccessibles.
- Construire un algorithme de diffusion qui informe tous les noeuds accessibles d'un réseau à topologie de grille rectangulaire arbitraire.
- Programmer une application démontrant le fonctionnement de l'algorithme ainsi que le résultat final.
- Estimer l'efficacité de l'algorithme en pratique.

Tous ces objectifs qui avaient été fixés dans le projet de maîtrise ont été atteints et présentés dans ce mémoire. Nous avons étudié, analysé et proposé une solution pour la diffusion dans les réseaux radio de type grille rectangulaire. Nous nous sommes intéressés à une seule classe de topologies (les grilles rectangulaires) ; un problème ouvert pourrait être de trouver un algorithme pour d'autres topologies de réseaux radio, par exemple, les grilles hexagonales qui sont à la base de la téléphonie cellulaire.

Dans notre étude, nous avons supposé que le degré des noeuds était un paramètre qu'il était possible de connaître et que chacun des noeuds connaissait son degré. Par contre, dans la réalité les téléphones cellulaires ou les stations de travail dans un réseau radio ne sont pas souvent au courant de leur degré. Il pourrait être intéressant de trouver un algorithme de diffusion dans lequel ce paramètre n'est pas connu par les noeuds.

Annexe A

Le code source partiel de l'application (grille.java)

```
package algo2;
import java.util.*;
import java.awt.*;
import javax.swing.*;

/* Voici le code source de la classe grille. Nous avons réduit au
maximum le code source de cette classe pour l'inclure dans ce
mémoire. Plusieurs fonctions ne sont pas dans cette annexe,
seules les fonctions directement liées à l'algorithme s'y
trouvent. Les classes sommet, arete, message, transmission,
application ainsi que tout ce qui se rapporte à l'affichage
graphique ont été exclues de ce mémoire. */

/**
 * <p>Title: Grille</p>
 * <p>Description: La classe qui représente la grille et l'algorithme</p>
 * <p>Copyright: Copyright (c) 2006</p>
 * @author Katia Larrivée
 * @version 2.5
 */

public class Grille {
  private static final int sizeVectorInit = 20;
  double d = Math.random();
  public int colonnes = (int) (d * 15);
  double s = Math.random();
```

```
public int lignes = (int) (s * 13);
public int nbsommets = colonnes * lignes;
public Vector sommets = null;
public Vector aretes = null;
public Vector aretesCopy = null;
public Vector sommetsCopy = null;
public Vector transmissions = null;
public int rondes_continuelles = 0;
public int voisins[]; // tableau des degres
public boolean arcs[][]; // tableau des arcs
public int listeVoisins[][] = null; // les numéros des voisins
public int rondes = 0; //le numéro de la ronde actuelle
public int source = -1; //le numéro de la source
public int ronde_commune = -1;
public int diametre = 0;

public int chef1_num = -1;
public int chef1_ID = -1;
public int chef2_num = -1;
public int chef2_ID = -1;
public int chef3_num = -1;
public int chef3_ID = -1;
public int chef4_num = -1;
public int chef4_ID = -1;
public int chef_balayage1 = -1;
public int chef_balayage2 = -1;

public Color vecteurCouleur[];
JTable jTable1 = new JTable();
JButton jButton1 = new JButton();

public int messages_election = 0;
public int messages_synchronisation = 0;
public int messages_balayage = 0;
public int messages_perdus = 0;
public int total_messages = 0;
public int noeuds_non_informes = 0;
public int noeuds = 0;
public boolean deuxieme_rondes = false;

public int transmissions_2_egaux = 0;
public int transmissions_2_inegaux = 0;
public int transmissions_3 = 0;
```

```
public int transmissions_3_egaux = 0;
public int transmissions_4_4_egaux = 0;
public int transmissions_4_2_egaux_coins = 0;
public int transmissions_4_2_egaux_face = 0;
public int transmissions_4_1_different = 0;
public int max_transmissions = 0;

public boolean impossible = false;
public boolean connue = false;
public boolean inconnue = false;


/**
 * diffusion_avec_source_connue
 *
 * @return Grille
 */
public Grille diffusion_avec_source_connue() {

  distance();

  if( ((Sommet) sommets.get(source)).getDegre() == 2){

    if(lignes == colonnes){
      source2_2_axes_egaux();
    }

    else {
      source2_2_axes_inegaux();
    }
  }

  else if (((Sommet) sommets.get(source)).getDegre() == 3){

    if(verification_demi_axes_egaux(((Sommet) sommets.get(source)).
    getID(), this.colonnes, this.lignes)){
      source3_egaux();
    }

    else{
      source3();
    }
  }
```

```
else {                    // source degre 4

  if (source == ( ( (lignes * colonnes) - 1) / 2) &&
      lignes == colonnes) {
    source4_4_axes_egaux();
  }

else {

  selection_axes_internes();
  connue = true;
  election_chefs();

  if (nombre_different(chef1_num, chef4_num, chef2_num, chef3_num)){
    source4_1_axe_different();
  }

  else if (chef1_num == chef4_num && chef2_num == chef3_num &&
           chef1_num != chef2_num) {
    source4_2_axes_egaux_face();
  }

      else {
        source4_2_axes_egaux_coins();
      }
    }
  }
  return this;
}

/**
 * diffusion_avec_source_inconnue
 *
 * @return Grille
 */
public Grille diffusion_avec_source_inconnue() {
    backup();
inconnue = true;
distance();

if( ((Sommet) sommets.get(source)).getDegre() == 2){
```

```
source2_2_axes_egaux();
transmissions_2_egaux = transmissions.size();
informe_huitieme_fois();
effacer_contenu_noeuds_non_source();
rondes = 0;

source2_2_axes_inegaux();
transmissions_2_inegaux = transmissions.size() -
  transmissions_2_egaux;
  informe_deuxieme_fois();
  effacer_contenu_noeuds_non_source();
  rondes = 0;
}

if( ((Sommet) sommets.get(source)).getDegre() == 3){

  source3();
  transmissions_3 = transmissions.size();
  informe_troisieme_fois();
  effacer_contenu_noeuds_non_source();
  rondes = 0;

  source3_egaux();
    transmissions_3_egaux = transmissions.size() - transmissions_3;
    informe_neuvieme_fois();
    effacer_contenu_noeuds_non_source();
    rondes = 0;
  }

if( ((Sommet) sommets.get(source)).getDegre() == 4){
  source4_4_axes_egaux();
  transmissions_4_4_egaux = transmissions.size();
  informe_quatrieme_fois();
  effacer_chefs();
  effacer_chefs_balayage();
  effacer_contenu_noeuds_non_source();
  rondes = 0;

  source4_2_axes_egaux_coins();
  transmissions_4_2_egaux_coins = transmissions.size() -
  transmissions_4_4_egaux;
  informe_septieme_fois();
  effacer_contenu_noeuds_non_source();
```

```
rondes = 0;

source4_2_axes_egaux_face();
transmissions_4_2_egaux_face = transmissions.size() -
transmissions_4_4_egaux - transmissions_4_2_egaux_coins;
informe_sixieme_fois();
effacer_chefs_balayage();
effacer_chefs();
effacer_contenu_noeuds_non_source();
rondes = 0;

source4_1_axe_different();
transmissions_4_1_different = transmissions.size() -
transmissions_4_4_egaux - transmissions_4_2_egaux_coins -
transmissions_4_2_egaux_face;
informe_cinquieme_fois();
effacer_chefs_balayage();
effacer_chefs();
effacer_contenu_noeuds_non_source();
rondes = 0;
}

rondes = rondes_continuelles;
verification_sommets_sept_algorithmes();
max_transmissions(this.transmissions_2_egaux,
this.transmissions_2_inegaux, this.transmissions_3,
this.transmissions_4_1_different, this.transmissions_4_2_egaux_coins,
this.transmissions_4_2_egaux_face,this.transmissions_4_4_egaux,
this.transmissions_3_egaux);

return this;
}

/**
 * source2_2_axes_inegaux
 *
 * @return Grille
 */
public Grille source2_2_axes_inegaux() {

selection_axes_externes(1); // 1 signifie que le chef c est la source
retour_message_source();
election_chef_champion(2, 2); // plus petit et degre 2
```

```
    retour_message_chef();
    selection_noeuds_balayage(1); // plus petit
    diffusion_par_balayage(true);

    return this;
  }

  /**
   * source2_2_axes_egaux
   *
   * @return Grille
   */
  public Grille source2_2_axes_egaux() {

    selection_axes_externes(1);

    if(lignes != 2 && colonnes !=2){
      retour_message_chef();
    }
    diffusion_par_balayage(false);

    return this;
  }

  /**
   * source3 avec demi-axes differents
   *
   * @return Grille
   */
  public Grille source3() {
    selection_axes_externes(1);
    rondes_continuelles = rondes * 2;

    diffusion_par_balayage(false);

    return this;
  }

  /**
   * source3 avec demi-axes egaux
   *
   * @return Grille
   */
```

```
public Grille source3_egaux() {
selection_axes_externes(1);

if(rondes_continuelles != 0){
    rondes_continuelles = rondes * 2 + rondes_continuelles;
}

else{
    rondes_continuelles = rondes * 2;
  }

  this.transmissions_3 = this.nbreTransmissions();
  diffusion_par_balayage_sans_source(false);
  this.transmissions_3_egaux = this.nbreTransmissions();
  diffusion_ligne_colonne_interne_test();
  changement_rondes();

  return this;
}

  /**
   * source4_4_axes_egaux
   *
   * @return Grille
   */
  public Grille source4_4_axes_egaux() {
    selection_axes_internes();
    retour_message_source();
    diffusion_par_balayage(false);
    impossible = false;

    return this;
  }

  /**
   * source4_1_axe_different
   *
   * @return Grille
   */
  public Grille source4_1_axe_different() {
    if (connue != true) {
      selection_axes_internes();
    }
```

```
retour_message_source();
election_chef_different(); // plus petit et degre 3
retour_message_chef();


if(impossible == false){
    selection_noeuds_balayage_cas_special(1); // plus petit
    diffusion_par_balayage(true);
  }


  impossible = false;
  return this;

}

/**
 * source4_2_axes_egaux_coins
 *
 * @return Grille
 */
public Grille source4_2_axes_egaux_coins() {

sommetsCopy = (Vector) sommets.clone();


  if(connue != true){
    selection_axes_internes();
    retour_message_source();
  }
  annulation_synchro();
  election_chef_champion(1, 3); // plus petit et degre 3
  retour_message_source();
  selection_noeuds_balayage_en_coin(); // plus petit
  diffusion_par_balayage(true);

  informe_premiere_fois();
  effacer_chefs_balayage();
  deuxieme_rondes = true;

  verification_sommets();
```

```
  return this;
}

/**
 * source4_2_axes_egaux_face
 *
 * @return Grille
 */
public Grille source4_2_axes_egaux_face() {


  if(connue != true){
    selection_axes_internes();
  }

  annulation_synchro();
  retour_message_source();
  election_chef_champion(); // plus grand et degre 3

  if(connue == true){
    impossible = false;
  }

if(impossible == false){
  retour_message_source();

  for(int i = 0; i < this.nbreSommets(); i++){
    if (( (Sommet) sommets.get(i)).getID() == chef_balayage1){
      ( (Sommet) sommets.get(i)).setPlusPetitChampion(true);
      ( (Sommet) sommets.get(i)).setElu(true);
    }

    if ( ( (Sommet) sommets.get(i)).getID() == chef_balayage2) {
      ( (Sommet) sommets.get(i)).setPlusPetitChampion(true);
      ( (Sommet) sommets.get(i)).setElu(true);
    }
    ( (Sommet) sommets.get(i)).sommet_deux = true;

  }
selection_axes_externes(2);
selection_noeuds_balayage_externe();
diffusion_par_balayage(true);
```

```
    }

    impossible = false;
    return this;
  }

  /**
   * fonction qui permet de savoir si un noeud est
   * élu ou non (Le point 3 de la procédure élection des axes
   * internes)
   *
   * @param transmetteur Sommet
   * @param destinataire Sommet
   */
  public void election_interne(Sommet transmetteur, Sommet destinataire) {
    int degre = voisins[destinataire.getID()];

    if (degre == 4) {
      if (destinataire.getNumero() % 2 == 1) { // le numero est impair
        if (destinataire.file_message.size() >= 2) {
          AlgoDMessage msg1 = destinataire.file_message.pop();
          AlgoDMessage msg2 = destinataire.file_message.pop();
          if (msg1.num_ronde ==
              (msg2.num_ronde + 2)) {

            if(ligne_colonne_speciale(destinataire)){
              destinataire.setElu(true);
              destinataire.transmission_elu = true;
            }
        }
      }
    }
  }
  else { // numero est pair
    if (destinataire.file_message.size() >= 2) {
      AlgoDMessage msg1 = destinataire.file_message.pop();
      AlgoDMessage msg2 = destinataire.file_message.pop();
      if ( (transmetteur.getElu() == true)) {
        if ( (msg1.num_ronde == (msg2.num_ronde + 3)) &&
        (transmetteur.getNumero() == (destinataire.getNumero()
        - 1))) {
      destinataire.setElu(true);
      destinataire.transmission_elu = true;
    }
```

```
            }
            else {
              destinataire.file_message.push(msg2);
              destinataire.file_message.push(msg1);
            }
          }
        }
      }
      if (degre == 3) {
        if (transmetteur.getElu() == true &&
            ligne_colonne_speciale(destinataire)) {
          destinataire.setChefLigne(true);
        }
      }
    }

    /**
     * fonction qui permet de savoir si un noeud est
     * élu ou non (la procedure 2 : election des axes externes)
     *
     * @param transmetteur Sommet
     * @param destinataire Sommet
     */
    public void election_externe(Sommet transmetteur, Sommet destinataire) {
      int degre = voisins[destinataire.getID()];

      if (degre == 3) {
        if(transmetteur.getSource() == true){
          destinataire.setElu(true);
          destinataire.transmission_elu = true;
        }

        if(transmetteur.getElu() == true){
          destinataire.setElu(true);
          destinataire.transmission_elu = true;
        }
      }
      else if (degre == 2) {
        if(transmetteur.getElu() == true){
          destinataire.setChefLigne(true);
          destinataire.transmission_elu = true;
```

```
    }
  }
}

/**
 * fonction qui permet de savoir si un noeud est
 * élu ou non (la procedure 1 : election des axes internes)
 */
public void selection_axes_internes() {
  ( (Sommet) sommets.get(source)).setCompteur(1);
  ( (Sommet) sommets.get(source)).setInforme(true);
  ( (Sommet) sommets.get(source)).setJamaisInforme(false);
  ( (Sommet) sommets.get(source)).setMessage(true);
  transmission_message( ( (Sommet) sommets.get(source)), false,
  "message", 1);
  ( (Sommet) sommets.get(source)).setMessage(false);
  rondes++;
  rondes_continuelles ++;

   int max = this.max_colonnes_lignes();

  for (int i = 0; i < max; i++) {
    transmission_tous_voisins_une_ronde_interne();
    verification_message_pop();
    verification_est_informe();
    rondes++;
    rondes_continuelles ++;
  }
}

/**
 * procedure 2 : Élection des noeuds externes
 *
 * @param source_chef int
 */
public void selection_axes_externes(int source_chef) {

  // 1 c'est la source, 2 c'est plus petit champion, 3 c'est plus grand
  // champion
  if (source_chef == 1) {
    ( (Sommet) sommets.get(source)).setCompteur(1);
    ( (Sommet) sommets.get(source)).setInforme(true);
    ( (Sommet) sommets.get(source)).setMessage(true);
```

```
    ( (Sommet) sommets.get(source)).setJamaisInforme(false);
    transmission_message( ( (Sommet) sommets.get(source)), false,
    "message", 2);
    ( (Sommet) sommets.get(source)).setMessage(false);
    rondes++;
    rondes_continuelles ++;
  }
  if (source_chef == 2) {
    int source2 = 0;
    int source3 = 0;
    for(int i = 0; i < this.nbreSommets(); i++){
      if( ((Sommet) sommets.get(i)).getPlusPetitChampion() == true){
        if(source2 != 0 && source2 != i){
          source3 = i;
        }
        else{
          source2 = i;
        }
      }
    }
    ( (Sommet) sommets.get(source2)).setMessage(true);
    ( (Sommet) sommets.get(source2)).setJamaisInforme(false);
    transmission_message( ( (Sommet) sommets.get(source2)), false,
    "message", 2);
    ( (Sommet) sommets.get(source2)).setMessage(false);
    ( (Sommet) sommets.get(source3)).setMessage(true);
    ( (Sommet) sommets.get(source3)).setJamaisInforme(false);
    transmission_message( ( (Sommet) sommets.get(source3)), false,
    "message", 2);
    ( (Sommet) sommets.get(source3)).setMessage(false);

    rondes++;
    rondes_continuelles ++;
  }

  int max = distance_coin();

  for (int i = 1; i < max; i++) {
    transmission_tous_voisins_une_ronde_externe();
    verification_message_pop();
    verification_est_informe();
    rondes++;
```

```
      rondes_continuelles ++;
    }
  }

  /**
   * procedure 2 : Élection des noeuds externes point numéro 4
   */
  public void transmission_tous_voisins_une_ronde_externe() {
    for (int i = 0; i < sommets.size(); i++) {

      if ( ( (Sommet) sommets.get(i)).getMessage() == true &&
           ( (Sommet) sommets.get(i)).getCompteur() == rondes &&
           ( (Sommet) sommets.get(i)).getDegre() != 4) {

       if(( (Sommet) sommets.get(i)).getInforme() == false ||
          ( (Sommet) sommets.get(i)).sommet_deux == true){

         if ( ( (Sommet) sommets.get(i)).getChefLigne() == true ||
              ( (Sommet) sommets.get(i)).getElu() == true) {
           ( (Sommet) sommets.get(i)).compteur++;
           transmission_message( ( (Sommet) sommets.get(i)), false,
           "message",
                                 2);
             ( (Sommet) sommets.get(i)).setMessage(false);
           }
        }
       }

       else if ( ( (Sommet) sommets.get(i)).getInforme() == true) {
         ( (Sommet) sommets.get(i)).compteur++;
       }
    }
  }

  /**
   * transmission_tous_voisins_une_ronde_interne (procédure 1)
   */
  public void transmission_tous_voisins_une_ronde_interne() {
    for (int i = 0; i < sommets.size(); i++) {

      if ( ( (Sommet) sommets.get(i)).getMessage() == true &&
           ( (Sommet) sommets.get(i)).getCompteur() == rondes) {
```

```
    ((Sommet) sommets.get(i)).compteur++;
    transmission_message( ( (Sommet) sommets.get(i)), false,
    "message", 1);
    ((Sommet) sommets.get(i)).setMessage(false);
}

else if ( ( (Sommet) sommets.get(i)).getInforme() == true) {
    ( (Sommet) sommets.get(i)).compteur++;
    }

    if ( ( (Sommet) sommets.get(i)).getElu() == true && (((Sommet)
    sommets.get(i)).transmission_elu = true)) {
      if(rondes > ((Sommet) sommets.get(i)).getNumero()+1){
      }

      if ( rondes == ( (Sommet) sommets.get(i)).getNumero() + 3) {
        transmission_message( ( (Sommet) sommets.get(i)), false, "elu",
         1);
        ( (Sommet) sommets.get(i)).transmission_elu = false;
      }
    }
  }
}

/**
 * transmission_message
 *
 * @param expediteur Sommet
 * @param source boolean
 * @param contenu String
 * @param parametre int
 */
public void transmission_message(Sommet expediteur, boolean source,
                                 String contenu, int parametre) {

  // si parametre egal 1 alors election interne
  // si parametre egal 2 alors election externe
  // si parametre egal 3 alors pas d'election
  int nbvoisins = getNbVoisins(expediteur.getID());

  for (int i = 0; i < nbvoisins; i++) {
    int num = listeVoisins[expediteur.getID()][i];
```

```
AlgoDMessage message = new AlgoDMessage(contenu, expediteur.getID(),
false,expediteur.getCompteur());

addTransmission(expediteur.getID(), num, rondes);
if(contenu == "message"){
  ((Transmission) transmissions.get(transmissions.size()-1)).
  election = true;
}

if (contenu == "message2") {
  ( (Transmission) transmissions.get(transmissions.size()-1)).
  election = true;
}

if (contenu == "synchro") {
  ( (Transmission) transmissions.get(transmissions.size()-1)).
  synchronisation = true;
}

if (contenu == "balayage") {
  ( (Transmission) transmissions.get(transmissions.size()-1)).
  balayage = true;
}

if (contenu == "interne") {
  ( (Transmission) transmissions.get(transmissions.size() - 1)).
  balayage = true;
}


verification_transmissions(num, rondes, expediteur.getID());

if(parametre == 3 &&
   (((Sommet) sommets.get(num)).getChefLigne()== true
    || ((Sommet) sommets.get(num)).getElu() == true
    || ((Sommet) sommets.get(num)).getInforme() == true)){
}

else if (((Sommet) sommets.get(num)).file_message.empty() == false){
AlgoDMessage msg = ((Sommet) sommets.get(num)).file_message.peek();

  if (msg.num_ronde == message.num_ronde) {
  //le noeud a recu plus d'un msg durant la ronde
```

```
AlgoDMessage msg2 = ((Sommet)sommets.get(num)).file_message.pop();
  msg2.msg_pop = true;
  ( (Sommet) sommets.get(num)).file_message.push(msg2);
}

else {
  ( (Sommet) sommets.get(num)).file_message.push(message);
  if ( ( (Sommet) sommets.get(num)).sommet_deux == true) {
  ( (Sommet) sommets.get(num)).file_message.push(message);
  ( (Sommet) sommets.get(num)).setMessage(true);
  ( (Sommet) sommets.get(num)).setNumero(expediteur.getCompteur());
  ( (Sommet) sommets.get(num)).setCompteur(rondes + 1);

  if(parametre == 1){
    election_interne(expediteur, ( (Sommet) sommets.get(num)));
  }

  if (parametre == 2){
    election_externe(expediteur, ( (Sommet) sommets.get(num)));
  }

}

  else{
    if (parametre == 1) {
      election_interne(expediteur, ( (Sommet) sommets.get(num)));
    }

    if (parametre == 2) {
      election_externe(expediteur, ( (Sommet) sommets.get(num)));
    }
  }
  }
}
else { // seul msg de cette ronde
  if ( ( (Sommet) sommets.get(num)).getInforme() == false) {
  ( (Sommet) sommets.get(num)).file_message.push(message);
  ( (Sommet) sommets.get(num)).setMessage(true);
  ( (Sommet) sommets.get(num)).setNumero(expediteur.
getCompteur());
  ( (Sommet) sommets.get(num)).setCompteur(expediteur.
getCompteur());
```

```
if(parametre == 1){
  election_interne(expediteur, ( (Sommet) sommets.get(num)));
}

if (parametre == 2){
  election_externe(expediteur, ( (Sommet) sommets.get(num)));
}
    }

    else {
      ( (Sommet) sommets.get(num)).file_message.push(message);

      if(parametre == 1){
        election_interne(expediteur, ( (Sommet) sommets.get(num)));
      }

      if (parametre == 2){
        election_externe(expediteur, ( (Sommet) sommets.get(num)));
      }
    }
  }
}


/**
 * diffusion_ligne_colonne_interne
 * La procedure 6
 */
public void diffusion_ligne_colonne_interne() {
  transmission_message( ( (Sommet) sommets.get(source)), false,
  "balayage", 3);
  ( (Sommet) sommets.get(source)).setMessage(false);
  rondes++;
  rondes_continuelles ++;

int max = max_colonnes_lignes();

for (int i = 0; i < max; i++) {
  transmission_tous_voisins_degres_4();
  verification_message_pop();
```

```
    verification_est_informe();
    rondes++;
    rondes_continuelles ++;

    if (i > 1) {
      incrementation_compteur(); // ??? à vérifier
    }
  }

  /**
   * transmission_tous_voisins_degres_4
   * La procedure 1
   */
  public void transmission_tous_voisins_degres_4() {
    for (int i = 0; i < sommets.size(); i++) {
      if ( ( (Sommet) sommets.get(i)).getInforme() == true) {
        ( (Sommet) sommets.get(i)).compteur++;
      }
      if ( ( (Sommet) sommets.get(i)).getDegre() == 4) {
        if ( ( (Sommet) sommets.get(i)).getMessage() == true &&
            ( (Sommet) sommets.get(i)).getCompteur() == rondes) {
          AlgoDMessage msg1 = ( (Sommet) sommets.get(1)).file_message.
          peek();
          if (msg1.contenu != "message2") {
            transmission_message( ( (Sommet) sommets.get(i)), false,
            "message2", 3);
            ( (Sommet) sommets.get(i)).setMessage(false);
          }
        }
      }
    }
  }


  /**
   * synchronisation des chefs de ligne (procédure 5)
   * reçoit 1 si chef de ligne, 2 si chef champion petit,
   * 3 si chef champion grand
   * @param chef int
   */
  public void synchronisation_chef(int chef) {
```

```
for (int i = 0; i < sommets.size(); i++) {

  if (chef == 1) {
    if ( ( (Sommet) sommets.get(i)).getChefLigne() == true) {
      ( (Sommet) sommets.get(i)).compteur++;
      ( (Sommet) sommets.get(i)).setRondeCommune( ( (Sommet)
      sommets.get(i)).getNumero());
      ( (Sommet) sommets.get(i)).setMessage(true);
      transmission_message( ( (Sommet) sommets.get(i)), true,
                            "synchro", 3);
      ( (Sommet) sommets.get(i)).setMessage(false);
      ronde_commune = ( (Sommet) sommets.get(i)).getNumero() + rondes;
      ( (Sommet) sommets.get(i)).setRondeCommune(ronde_commune);
    }
  }

  if (chef == 2) {
    if ( ( (Sommet) sommets.get(i)).getPlusPetitChampion() == true) {
      ( (Sommet) sommets.get(i)).compteur++;
      ( (Sommet) sommets.get(i)).setRondeCommune( ( (Sommet)
      sommets.get(i)).getNumero());
      ( (Sommet) sommets.get(i)).setMessage(true);
      transmission_message( ( (Sommet) sommets.get(i)), true,
                            "synchro", 3);
      ( (Sommet) sommets.get(i)).setMessage(false);
      ronde_commune = ( (Sommet) sommets.get(i)).getNumero() + rondes;
      ( (Sommet) sommets.get(i)).setRondeCommune(ronde_commune);
    }
  }

  if (chef == 3) {
    if ( ( (Sommet) sommets.get(i)).getPlusGrandChampion() == true) {
      ( (Sommet) sommets.get(i)).compteur++;
      ( (Sommet) sommets.get(i)).setRondeCommune( ( (Sommet)
      sommets.get(i)).getNumero());
      ( (Sommet) sommets.get(i)).setMessage(true);
      transmission_message( ( (Sommet) sommets.get(i)), true,
                            "synchro", 3);
      ( (Sommet) sommets.get(i)).setMessage(false);
      ronde_commune = ( (Sommet) sommets.get(i)).getNumero() + rondes;
      ( (Sommet) sommets.get(i)).setRondeCommune(ronde_commune);
    }
```

```
}

rondes++;
rondes_continuelles ++;
int max = max_colonnes_lignes();

for (int j = 0; j < max; j++) {
  transmission_tous_voisins_une_ronde_synchronisation(chef);
      verification_message_pop();
      verification_est_informe();
      rondes++;
      rondes_continuelles ++;

      if (j > 1) {
        incrementation_compteur();
      }
    }
  }
}

/**
 * retour_message_source procédure 3
 */
public void retour_message_source() {
  int max = distance_coin() -1;

  for (int j = 0; j < max; j++) {
    rondes_continuelles ++;
  }
}

/**
 * diffusion_par_balayage Procédure 7
 *
 * @param synchro boolean
 */
public void diffusion_par_balayage(boolean synchro) {
  int max = max_colonnes_lignes();

effacer_contenu_noeuds_non_elus();

if(synchro == true){
  effacer_contenu_noeuds_non_synchro();
```

```
}

for (int i = 0; i < sommets.size(); i++) {
  if(((Sommet) sommets.get(i)).getElu() == true ||
  ((Sommet) sommets.get(i)).getChefLigne() == true){
    if((synchro == true && ((Sommet) sommets.get(i)).synchro == true)
    || (synchro == false)){
      transmission_message( ( (Sommet) sommets.get(i)), false,
      "balayage", 3);
      ( (Sommet) sommets.get(i)).compteur++;
    }
  }
}

verification_message_pop();
verification_est_informe();
rondes++;
rondes_continuelles ++;

for (int j = 0; j < max; j++) {
  transmission_tous_voisins_une_ronde_balayage();
  verification_message_pop();
  verification_est_informe();
  rondes++;
  rondes_continuelles ++;
  }

  if (synchro == true) {
    effacer_contenu_noeuds_synchro();
  }

}

/**
 * election_chefs_coins_speciaux Procédure 8
 */
public void election_chefs_coins_speciaux(){
  for (int i = 0; i < sommets.size(); i++) {

if ( ( (Sommet) sommets.get(i)).getSource() == true) {
  ( (Sommet) sommets.get(i)).compteur++;
  ( (Sommet) sommets.get(i)).setMessage(true);
  transmission_message( ( (Sommet) sommets.get(i)), true,
```

```
                                "election", 3);
      ((Sommet) sommets.get(i)).setMessage(false);
    }
  }

  rondes++;
  rondes_continuelles ++;
  int max = max_colonnes_lignes();
    for (int j = 0; j < max; j++) {
      transmission_tous_voisins_une_ronde_externe();
      verification_message_pop();
      verification_est_informe();
      rondes++;
    }
  }

  /**
   * election_chef_champion Procédure 4
   *
   * @param grand_petit int
   * @param degre int
   * @return int
   */
  public int election_chef_champion(int grand_petit, int degre) {

    // int grand_petit == 1, election du plus grand
    // int grand_petit == 2, election du plus petit
    // int degre == 2, d'un coin
    // int degre == 3, d'un noeud de cote
    // boolean cas_special == true nous sommes dans le cas
    // special ou nous voulons élire un noeud de degre 3 seulement

    if (degre == 2) {
      for (int i = 0; i < sommets.size(); i++) {

        if ( ((Sommet) sommets.get(i)).getChefLigne() == true &&
            ((Sommet) sommets.get(i)).getSource() == false &&
            ((Sommet) sommets.get(i)).getDegre() == 2) {
  if (chef1_num == -1) {
    chef1_num = ((Sommet) sommets.get(i)).getNumero();
    chef1_ID = ((Sommet) sommets.get(i)).getID();
  }
```

```
        else {
          chef2_num = ( (Sommet) sommets.get(i)).getNumero();
          chef2_ID = ( (Sommet) sommets.get(i)).getID();
        }
      }
    }

    if (chef1_ID != -1 && chef2_ID != -1) {
      if (grand_petit == 1) {
        if (chef1_num > chef2_num) {
          ( (Sommet) sommets.get(chef1_ID)).setPlusGrandChampion(true);
          ( (Sommet) sommets.get(chef2_ID)).setChefLigne(false);
          return chef1_ID;
        }

        if (chef2_num > chef1_num) {
          ( (Sommet) sommets.get(chef2_ID)).setPlusGrandChampion(true);
          ( (Sommet) sommets.get(chef1_ID)).setChefLigne(false);
          return chef2_ID;
        }

        else {
          return -1;
        }
      }
      if (grand_petit == 2) {
        if (chef1_num < chef2_num) {
          ( (Sommet) sommets.get(chef1_ID)).setPlusPetitChampion(true);
          ( (Sommet) sommets.get(chef2_ID)).setChefLigne(false);
          return chef1_ID;
        }

        if (chef2_num < chef1_num) {
          ( (Sommet) sommets.get(chef2_ID)).setPlusPetitChampion(true);
          ( (Sommet) sommets.get(chef1_ID)).setChefLigne(false);
          return chef2_ID;
        }

        else {
          return -1;
        }
      }
    }
```

```
    }
  }
  else {                                    // degres 3
    for (int i = 0; i < sommets.size(); i++) {

      if ( ( (Sommet) sommets.get(i)).getChefLigne() == true &&
           ( (Sommet) sommets.get(i)).getSource() == false &&
           ( (Sommet) sommets.get(i)).getDegre() == 3) {

        if ( ( (Sommet) sommets.get(i)).getChefLigne() == true &&
             ( (Sommet) sommets.get(i)).getSource() == false) {
          if (chef1_num == -1) {
            chef1_num = ( (Sommet) sommets.get(i)).getNumero();
            chef1_ID = ( (Sommet) sommets.get(i)).getID();
          }

          else if (chef2_num == -1) {
            chef2_num = ( (Sommet) sommets.get(i)).getNumero();
            chef2_ID = ( (Sommet) sommets.get(i)).getID();
          }
          else if (chef3_num == -1) {
            chef3_num = ( (Sommet) sommets.get(i)).getNumero();
            chef3_ID = ( (Sommet) sommets.get(i)).getID();
          }
          else if (chef4_num == -1) {
            chef4_num = ( (Sommet) sommets.get(i)).getNumero();
            chef4_ID = ( (Sommet) sommets.get(i)).getID();
          }
        }
      }
    }
  }

  if (chef1_ID != -1 && chef2_ID != -1 && chef3_ID != -1 &&
  chef4_ID != -1) {

    if (grand_petit == 1) {

      if (chef1_num > chef2_num) {
      ( (Sommet) sommets.get(chef1_ID)).setPlusGrandChampion(true);
      chef_balayage1 = chef1_ID;
      ( (Sommet) sommets.get(chef3_ID)).setPlusGrandChampion(true);
      chef_balayage2 = chef3_ID;
      return chef1_ID;
```

```
}

if (chef1_num < chef2_num) {
  ( (Sommet) sommets.get(chef2_ID)).setPlusGrandChampion(true);
  chef_balayage1 = chef2_ID;
  ( (Sommet) sommets.get(chef4_ID)).setPlusGrandChampion(true);
  chef_balayage2 = chef4_ID;
  return chef2_ID;
}

if (chef1_num > chef3_num) {
  ( (Sommet) sommets.get(chef2_ID)).setPlusGrandChampion(true);
  chef_balayage1 = chef2_ID;
  ( (Sommet) sommets.get(chef1_ID)).setPlusGrandChampion(true);
  chef_balayage2 = chef1_ID;
  return chef1_ID;
}

if (chef1_num < chef3_num) {
  ( (Sommet) sommets.get(chef3_ID)).setPlusGrandChampion(true);
  chef_balayage1 = chef3_ID;
  ( (Sommet) sommets.get(chef4_ID)).setPlusGrandChampion(true);
  chef_balayage2 = chef4_ID;
  return chef3_ID;
  }

  else {
  return -1;
  }
}

if (grand_petit == 2) {
  if (chef1_num < chef2_num) {
    ( (Sommet) sommets.get(chef1_ID)).setPlusPetitChampion(true);
    chef_balayage1 = chef1_ID;
    ( (Sommet) sommets.get(chef3_ID)).setPlusPetitChampion(true);
    chef_balayage2 = chef3_ID;
    return chef1_ID;
}

if (chef1_num > chef2_num) {
  ( (Sommet) sommets.get(chef2_ID)).setPlusPetitChampion(true);
  chef_balayage1 = chef2_ID;
```

```
        ((Sommet) sommets.get(chef4_ID)).setPlusPetitChampion(true);
        chef_balayage2 = chef4_ID;
        return chef2_ID;
    }

    if (chef1_num < chef3_num) {
        ((Sommet) sommets.get(chef2_ID)).setPlusPetitChampion(true);
        chef_balayage1 = chef2_ID;
        ((Sommet) sommets.get(chef1_ID)).setPlusPetitChampion(true);
        chef_balayage2 = chef1_ID;
        return chef1_ID;
    }

    if (chef1_num > chef3_num) {
        ((Sommet) sommets.get(chef3_ID)).setPlusPetitChampion(true);
        chef_balayage1 = chef3_ID;
        ((Sommet) sommets.get(chef4_ID)).setPlusPetitChampion(true);
        chef_balayage2 = chef4_ID;
        return chef3_ID;
    }

    else {
        return -1;
                }
            }
        }
    }
    return -1;
  }
}
```

Bibliographie

[1] ALON, N., BAR-NOY, A., LINIAL, N., PELEG, D. A Lower Bound for Radio Broadcast. *Journal of Computer and System Sciences 43* (1991), 290-298.

[2] BAR-YEHUDA, R., GOLDREICH, O., ITAI, A. On the Time-Complexity of Broadcast in Multi-hop Radio Networks : An Exponential Gap between Determinism and Randomization. *Journal of Computer and System Sciences 45* (1992), 104-126.

[3] BRUSCHI, D., DEL PINTO, M. Lower bounds for the broadcast problem in mobile radio networks. *Distributed Computing 10* (1997), 129-135.

[4] CHLAMTAC, I., KUTTEN, S. On broadcasting in Radio Networks - Problem Analysis and Protocol Desigh. *IEEE Transactions on Communications 33* (1985), 1240-1246.

[5] CHLAMTAC, I., WEINSTEIN, O. The Wave Expansion Approach to Broadcasting in Multihop Radio Networks. *IEEE Transactions on Communications 39* (1991), 426-433.

[6] CHLEBUS, S., GASIENIEC, L., OSTLIN, A., ROBSON, J. M. Deterministic Radio Broadcasting. In *Proceedings 27th Int. Colloquium on Automata, Languages and Programming* (2000), 717-728.

[7] CHLEBUS, S., GASIENIEC, L., GIBBONS, A., PELC, A., RYTTER, W. Deterministic broadcasting in unknown radio networks. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms* (2000), 861-871.

[8] CHROBAK, M., GASIENIEC, L., RYTTER, W. Fast Broadcasting and Gossiping in Radio Networks. *IEEE Symposium on Foundations of Computer Science* (2000), 575-581.

[9] CLEMENTI, A., MONTI A., SILVESTRI R. Selective Families, Superimposed Codes, and Broadcasting in Unknown Radio Networks. *12th ACM-SIAM Symposium On Discrete Algorithms (SODA 2001)* (2001), 709-718.

[10] CZUMAJ, A., RYTTER, W. Broadcasting Algorithms in Radio Networks with Unknown Topology. In *Proceedings 44th IEEE Symposium on Foundations of Computer Science (FOCS 2003)* (2003), 492-501.

[11] DIAZ OLAVARRIETA, L., APARICIO NAVA A. Wireless Communications : A Bird's Eye View Of an Emerging Technology. *International Symposium on Communications and Information Technologies (ISCIT 2004)* (2004), 541-546.

[12] ELKIN, M., KORTSARZ, G. Polylogarithmic Inapproximability of the Radio Broadcast Problem Approximation, Randomization, and Combinatorial Optimization, Algorithms and Techniques. *7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2004)* (2004), 105-116.

[13] GABER, I., MANSOUR, Y. Broadcast in Radio Networks. In *Proceedings 6th ACM-SIAM Symposium on Discrete Algorithms* (1995), 577-585.

[14] GASIENIEC, L., PELEG, D., XIN, Q. Faster Communication in Known Topology Radio Networks. In *Proceedings 24th ACM Symposium on Principles of Distributed Computing* (2005), 129-137.

[15] KOWALSKI, D., PELC, A. Centralized deterministic broadcasting in undirected multi-hop radio networks. *7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2004)* (2004), 171-182.

[16] KOWALSKI, D., PELC, A. Faster deterministic broadcasting in ad hoc radio networks *SIAM Journal on Discrete Mathematics 18* (2004), 332-346.

[17] KUSHILEVITZ, E., MANSOUR, Y. An $\Omega(Dlog(N/D))$ lower bound for broadcast in radio networks. *SIAM Journal of Computer 27* (1998), 702-712.

[18] LI, H., LOTT M., WECKERIE M., ZIRWAS W., SCHULZ E. Multihop communication in future mobile radio networks. In *Proceedings 44th IEEE Symposium on Foundations of Computer Science* (2002), 54-58.

[19] PELC, A. Waking up anonymous ad hoc radio networks. *19th International Symposium on Distributed Computing (DISC 2005)* (2005), 260-272.