Université de Montréal

Multicast explicite dans les réseaux ad hoc mobiles : implémentation, analyse et simulations d'un nouveau protocole multicast pour MANETs

par

Cédric FERRARIS

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de
Maître ès Sciences (M.Sc.)
en informatique

Mars, 2007

Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé :

Multicast explicite dans les réseaux ad hoc mobiles : implémentation, analyse et simulations d'un nouveau protocole multicast pour MANETs

présenté par :

Cédric FERRARIS

a été évalué par un jury composé des personnes suivantes :

Julie Vachon
président-rapporteur

Abdelhakim Hafid
directeur de recherche

Houari Sahraoui
membre du jury

Mémoire accepté le 20 avril 2007

Résumé

Le multicast est la transmission de paquets à un groupe d'hôtes identifiés par une seule adresse IP (adresse multicast). Il permet une utilisation efficace de la bande passante et a de nombreuses applications telles que la vidéoconférence et la distribution de contenu. Un réseau ad hoc mobile (MANET) est un réseau sans fil sans infrastructure ou administration centralisée. Les changements fréquents et imprévisibles dans la topologie du réseau (dus à la mobilité des nœuds) rendent l'utilisation de protocoles multicast existants dans les réseaux filaires non appropriée. Plusieurs protocoles multicast ont donc été proposés à ce jour pour faire face à ce problème mais aucun d'entre eux ne fait figure de standard. Mon projet de recherche consiste d'une part à valider et vérifier un protocole multicast pour MANETs appelé EM2NET et adapté aux petits groupes multicast. En effet, nous nous focalisons dans ce mémoire sur les protocoles adaptés à des environnements de faible densité, connus sous le nom de protocoles multicast explicites. Puis il s'agit de l'implémenter sous le logiciel de simulation de réseaux ns-2 et, au travers d'une analyse théorique et de simulations avancées, de mener une évaluation de performances et de le comparer avec un certain nombre de protocoles multicast pour MANETs déjà proposés dans la littérature (ODMRP, MAODV et E2M). Dans une deuxième partie, nous proposons des solutions et méthodes afin d'améliorer l'efficacité de EM2NET. Les résultats indiquent qu'il offre de meilleures performances que les méthodes de multicast explicite existantes. De plus, lorsqu'on le compare avec un protocole basé sur une structure de maillage, il offre une meilleure conservation d'énergie, métrique particulièrement importante dans les réseaux ad hoc. Enfin, la troisième et dernière contribution consiste en la proposition d'un protocole hybride basé sur l'utilisation en alternance des protocoles EM2NET et ODMRP.

Mots-clés : Multicast, réseaux ad hoc, protocoles multicast explicites, petits groupes multicast, implémentation, simulation, ns-2, évaluation de performances, comparaisons, conservation d'énergie.

Abstract

Multicasting is a very useful mechanism which is intended for group-oriented communication and allows reducing bandwidth utilization. It consists in transmitting datagrams to a group of hosts identified by a single multicast address. A mobile ad hoc network (MANET) is a dynamically reconfigurable wireless network with no fixed infrastructure or central administration. Due to nodes mobility in MANETs, the network topology changes frequently and unpredictably. These constraints make routing and multicasting in ad hoc networks extremely challenging. During recent years, several multicast protocols have been proposed specifically for MANETs but none of them is a standard. My research project consists first in verifying and validating a new multicast protocol for MANETs called EM2NET; this protocol is well suited for small multicast groups. Indeed, we focus in this dissertation on protocols designed for low density environments, known as explicit multicast protocols. Then, we implement it using the network simulator ns-2 and, through an analytic analysis and extensive simulations, we evaluate its performance and compare it with existing multicast protocols (ODMRP, MAODV and E²M). In a second part, we propose improvements to EM2NET in order to enhance its efficiency. Results indicate that EM2NET outperforms existing explicit multicast methods and tree-based schemes in ad hoc networks. In addition, when compared to a mesh-based protocol, we show that it leads to better energy conservation, which is an important parameter in MANETs. Finally, the third contribution is the proposition of a hybrid multicast protocol. The basic idea behind this protocol is to switch, in real-time, between ODMRP and EM2NET based on the state of the multicast group (group size, mobility speed, multicast traffic). Simulations show that the hybrid protocol outperforms existing protocols in terms of energy savings and reliability.

Keywords: Multicast, ad hoc networks, explicit multicast protocols, small multicast groups, implementation, simulations, ns-2, performance evaluation, comparisons, energy conservation.

Table des matières

Liste des figures

Liste des tableaux

Liste des sigles et abréviations

AMRIS	Ad hoc Multicast Routing protocol utilizing Increasing id-numberS
AMRoute	Ad hoc Multicast Routing
AODV	Ad hoc On-demand Distance Vector
CAMP	Core-Assisted Mesh Protocol
DDM	Differential Destination Multicast
DSR	Dynamic Source Routing
E²M	Extended Explicit Multicast
EM2NET	Explicit Multicast for MANETs
FG	Forwarding Group
IETF	Internet Engineering Task Force
IN	Intercepting Node
MACT	Multicast ACTivation
MANET	Mobile Ad hoc NETworks
MAODV	Multicast Ad hoc On-demand Distance Vector
MFIT	Multicast Forwarding IN Table
Msm-id	Multicast session member id
Ns-2	Network Simulator 2
ODMRP	On-Demand Multicast Routing Protocol
RREQ/RREP	Route Request / Route Response
RTS/CTS	Request To Send / Clear To Send
SGM	Small Group Multicast
Sid	Smallest-id
TTL	Time-To-Live
WiFi	Wireless Fidelity
Xcast	Explicit Multicast
XF	Xcast Forwarder

Chapitre 1

Introduction

1.1 Définition du multicast

Le multicast est la transmission de paquets à un groupe d'hôtes identifiés par une seule adresse IP (adresse multicast). C'est un procédé de routage des données qui consiste à envoyer un même paquet de données à plusieurs personnes simultanément, allégeant ainsi de beaucoup la charge sur le réseau. En effet, plutôt que de devoir initier *n* transmissions unicast, on initie 1 transmission multicast vers *n* destinations. Il permet donc une utilisation efficace de la bande passante et trouve des applications dans de nombreux domaines (la vidéoconférence, la distribution de contenu, les applications collaboratives ou encore la mise à jour de parcs de machines). La figure 1.1 illustre l'efficacité du multicast en termes de ressources réseau lorsqu'il s'agit de communiquer avec plusieurs stations d'un même groupe. Une transmission multicast envoie un seul paquet qui sera dupliqué au besoin afin d'atteindre toutes les destinations tandis qu'une transmission unicast envoie autant de paquets qu'il y a de destinations.

Un groupe multicast est un ensemble de *n* machines. Il est entièrement dynamique (une station peut rejoindre ou quitter le groupe à tout moment) et ouvert (une station peut émettre un paquet dans un groupe sans en faire partie). Un groupe multicast est désigné par une adresse IP (adresse de classe D, de 224.0.0.1 à 239.255.255.255). Lorsqu'un poste veut envoyer un paquet à un groupe multicast, il envoie ce paquet à l'adresse IP identifiant ce groupe.

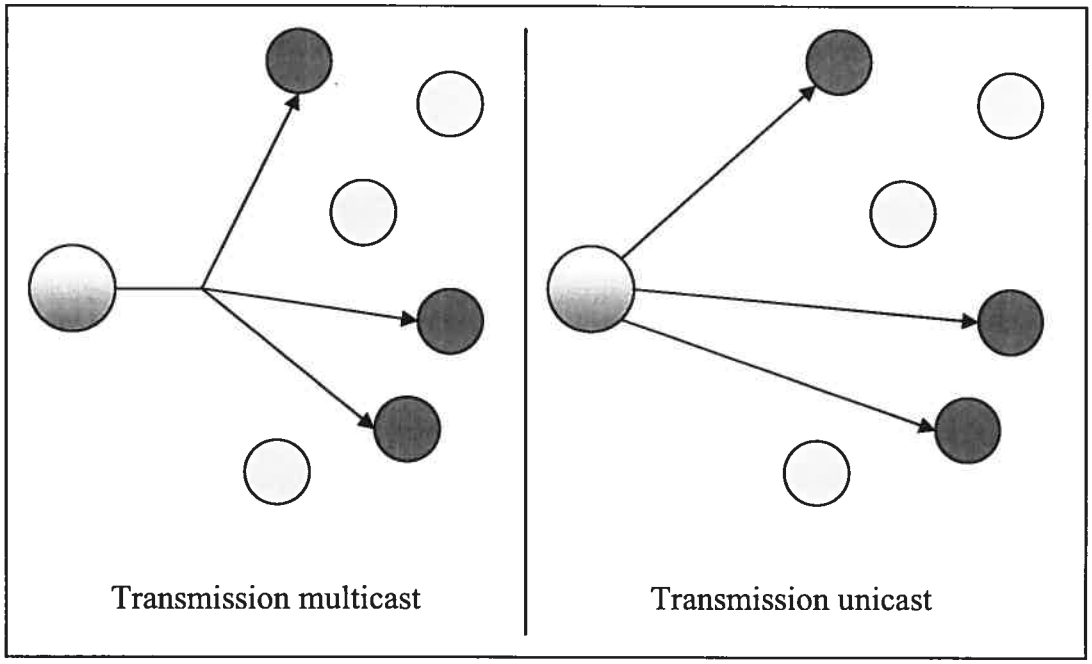

Figure 1.1. Différence entre une transmission multicast et unicast.

Il est intéressant d'évoquer l'existence du réseau MBone (*Multicast Backbone*, [1, 2]) qui est une implémentation du système de multicast IP permettant à la communauté des utilisateurs reliés au réseau internet de participer à des conférences audio/vidéo à partir de leur poste de travail. Mbone est un réseau virtuel de diffusion multipoint issu d'expérimentations effectuées en 1992 à l'IETF San Diego et visant à diffuser sur l'internet le son puis la vidéo de ses réunions plénières. L'idée était de créer une zone de test grande nature et quasi permanente. Le MBone s'appuie sur le réseau internet classique et forme une couche d'abstraction au dessus de celui-ci. Il est constitué de machines exécutant le démon de routage *mrouted* qui tissent une toile d'araignée constituée de tunnels au-dessus de l'internet (Cf. figure 1.2). En utilisant l'infrastructure des réseaux actuels, MBone évite le besoin d'avoir des réseaux dédiés pour la communication multimédia. Il y a à ce jour 1100 sous réseaux de diffusion MBone dans 26 pays. Il faut savoir que le MBone n'est pas la seule initiative du genre. Citons entre autres *Abilene Backbone Network* [3].

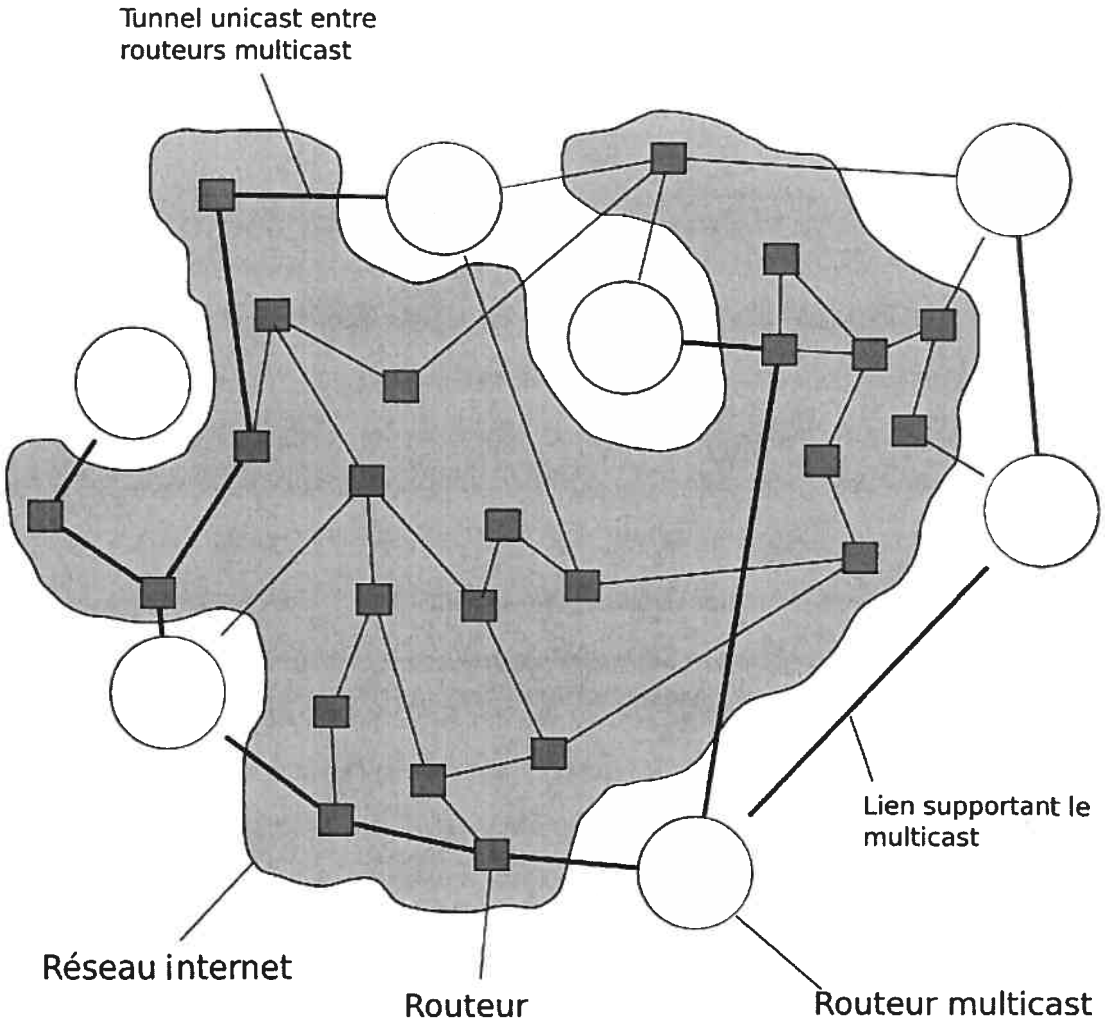

Figure 1.2. Architecture logique du réseau Mbone.

1.2 Introduction aux réseaux ad hoc

Un réseau ad hoc, à la différence des réseaux WiFi classiques, est un réseau sans infrastructure composé de nœuds mobiles. Ces terminaux mobiles agissent comme des routeurs (il n'y a pas d'administration centralisée) et ont pour but d'accomplir en commun une tâche précise. Ils communiquent par liaison radio, directement avec leurs voisins lorsqu'ils sont à portée de transmission l'un de l'autre ou par l'intermédiaire d'autres nœuds lorsque ce n'est pas le cas. Sa capacité à

fournir rapidement et de façon flexible des moyens de communication font d'un réseau ad hoc un choix idéal pour certaines applications personnelles, publiques ou d'entreprise. On peut par exemple citer les conférences et meetings en passant par les opérations militaires jusqu'aux situations d'urgence (catastrophes naturelles, ...) et la conduite coopérative entre véhicules. Les recherches autour des réseaux ad hoc portent actuellement sur le domaine militaire, commercial et universitaire.

Les avantages de tels réseaux sont la rapidité de mise en place, le coût faible et l'indépendance vis-à-vis de points d'accès. Cependant, il convient de relever certains points qui peuvent constituer un obstacle à leur développement. Tout d'abord, le caractère mobile des nœuds entraîne de fréquents changements dans la topologie du réseau. De plus, les nœuds sont souvent pourvus d'une batterie dont la capacité d'énergie est faible. Enfin, les communications sans fil sont soumises à une bande passante limitée et peuvent également être la cible de menaces de sécurité potentielles. Nous allons voir en quoi ces avantages et inconvénients influent sur la définition d'un protocole de routage multicast pour les réseaux ad hoc.

1.3 Problématique

La recherche d'un chemin d'accès entre deux nœuds du réseau est assurée par une fonction de routage. Cette fonction est d'autant plus complexe dans les réseaux ad hoc que les noeuds sont mobiles. Cela signifie que la topologie du réseau est constamment modifiée. Définir un protocole de routage multicast dans les réseaux ad hoc est donc un défi à part entière et utiliser les approches employées dans les réseaux filaires ne s'avère pas être une bonne solution : nous verrons pourquoi plus loin dans ce document. Des recherches ont ainsi été menées afin d'étendre les principes utilisés dans les réseaux filaires aux réseaux ad hoc [4]. Mais parmi les protocoles proposés, aucun d'entre eux ne fait figure de standard.

De plus, pour supporter le multicast dans un réseau, les nœuds intermédiaires doivent maintenir des informations d'état pour chacune des sessions multicast. Cette approche, lorsque appliquée à des groupes éparses peut s'avérer extrêmement coûteuse et souffrir de problèmes d'extensibilité. La plupart des schémas traditionnels sont en effet désignés pour supporter un petit nombre de larges groupes multicast. En plus de cela, les fréquents et imprévisibles changements dans la topologie d'un réseau ad hoc (dus à la mobilité des nœuds) rendent difficile le maintien d'informations correctes. Pour pallier ces problèmes, un nouveau schéma multicast a donc été proposé : le schéma Xcast (pour *Explicit Multicast*, [5, 6]). C'est un schéma spécialement adapté aux petits groupes multicast qui offre une meilleure extensibilité relativement au nombre de sessions. Le principe est que la source de la session encode explicitement dans l'en-tête du paquet de données la liste des destinations et délègue le processus de livraison des données au protocole unicast sous-jacent. Parmi les protocoles de cette catégorie, nous pouvons citer DDM [7] ou bien encore E²M [8]. Ces protocoles sont aussi classés dans la catégorie des protocoles multicast sans état (ou *stateless*) car les routeurs intermédiaires ne maintiennent pas d'information d'état sur les sessions multicast. Cependant, ils se heurtent au problème de la mobilité car ils ne proposent pas de mécanisme de reconstruction de l'arbre et/ou ne prennent pas en compte efficacement le déplacement des nœuds.

1.4 Contributions

De nombreuses études [9, 10] ont été menées afin de comparer les protocoles multicast dans les réseaux ad hoc. Cependant, aucune d'entre elles ne s'est employée à montrer l'influence de la densité du réseau sur les performances des protocoles. Nous avons vu qu'un nouveau schéma adapté aux petits groupes multicast a récemment été proposé. Parmi les protocoles de cette catégorie, nous allons présenter plus particulièrement l'un d'entre eux. Il s'agit de EM2NET [11] qui est un protocole

multicast explicite adapté aux petits groupes qui se propose d'outrepasser les limitations des protocoles actuels de cette catégorie. L'ensemble des travaux de recherche de ce mémoire sont basés sur EM2NET. La première contribution de ce document consiste à valider et implémenter le protocole EM2NET sous le logiciel de simulation de réseaux ns-2 (*Network Simulator*, [12]) et évaluer ses performances en le comparant avec un certain nombre de protocoles multicast pour les réseaux ad hoc déjà proposés dans la littérature, tels que ODMRP [13], MAODV [14] et E²M [8]. ODMRP et MAODV sont déjà disponibles dans l'environnement de simulation ns-2 mais ce n'est pas le cas de E²M qu'il a donc également fallu implémenter. Outre les simulations menées sous ns-2, l'évaluation des performances comprend aussi une analyse théorique qui compare certaines opérations et caractéristiques communes aux protocoles EM2NET et E²M. L'objectif est de montrer d'une part l'efficacité d'EM2NET lorsqu'on le compare avec des protocoles de sa catégorie et d'autre part, de mettre en avant la nécessité d'utiliser un protocole multicast explicite dans des environnements composés d'un grand nombre de petits groupes.

La seconde contribution a pour but d'identifier les points faibles du protocole EM2NET (en se basant sur les résultats des simulations) et de proposer des solutions permettant d'améliorer son efficacité. Ces solutions sont implémentées et validées à l'aide de simulations sous ns-2.

Enfin, la troisième contribution est la proposition d'un protocole hybride basé sur EM2NET et ODMRP. Le principe est le suivant : en se basant sur la vitesse de mobilité des nœuds et sur la taille du groupe multicast (qui sont des paramètres pouvant varier dans le temps), nous proposons un mécanisme d'alternance entre les protocoles ODMRP et EM2NET. Par exemple, lorsque l'environnement est favorable à EM2NET (c'est-à-dire qu'il est susceptible, d'après les simulations, d'obtenir les meilleures performances à cet instant t), le mode de fonctionnement du protocole hybride bascule vers EM2NET, et vice versa. Nous détaillons dans ce document les opérations nécessaires à l'implémentation de ce protocole hybride.

1.5 Structure du mémoire

Le reste du mémoire est organisé de la façon suivante. Dans le chapitre 2, nous traitons spécifiquement des réseaux ad hoc, de leurs caractéristiques et des applications dont ils font l'objet. Le chapitre 3 est consacré à un état de l'art des protocoles multicast proposés à ce jour pour les réseaux ad hoc. En conclusion de ce chapitre, nous identifions les avantages et inconvénients de chacun d'entre eux et montrons la nécessité d'un nouveau protocole multicast adapté aux petits groupes. Le chapitre 4 représente la plus grande partie du travail effectué. Nous présentons le protocole EM2NET, l'environnement de simulation, les détails de l'implémentation sous ns-2 et comparons ses performances avec d'autres protocoles multicast au moyen de simulations avancées et d'une analyse théorique. A l'aide des résultats obtenus, nous mettons en avant, dans le chapitre 5, les points à améliorer dans EM2NET, proposons des solutions et menons des simulations afin de prouver leur efficacité. Dans la seconde partie de ce chapitre, nous proposons une solution hybride combinant EM2NET et ODMRP qui consiste en une alternance entre les modes de fonctionnement de ces deux protocoles. Enfin, le chapitre 6 conclut ce document en identifiant les problèmes ouverts et les possibles travaux futurs.

Chapitre 2

Les réseaux ad hoc mobiles

2.1 Introduction

Les réseaux ad hoc (en latin : « qui va vers ce vers quoi il doit aller », c'est-à-dire « formé dans un but précis ») sont des réseaux sans fil ne nécessitant pas le besoin d'une infrastructure fixe pour communiquer. Dans leur configuration mobile, ils sont également connus sous le nom de MANETs (Mobile Ad hoc NETworks). L'ensemble des entités mobiles sont alors interconnectées par une technologie sans fil, formant ainsi un réseau temporaire, et coopèrent afin d'effectuer une tâche commune. Nous verrons plus loin dans ce chapitre quelles peuvent être les applications de tels réseaux. Au sein d'un réseau ad hoc, chacun des nœuds agit comme un routeur : il n'y a pas d'administration centralisée ou point d'accès comme dans les réseaux WiFi classiques. En effet, là où une ou plusieurs stations de base sont nécessaires à la plupart des communications entre les différents nœuds du réseau WiFi, les réseaux ad hoc s'organisent d'eux-mêmes et chaque entité peut jouer plusieurs rôles. Une entité communique directement avec sa voisine. Pour communiquer avec d'autres nœuds, il est nécessaire que les données transitent par des entités qui se chargeront de les acheminer. La figure 2.1 suivante illustre la différence entre un réseau avec infrastructure (un point d'accès est nécessaire afin de communiquer entre A et B) et un réseau ad hoc (la communication se fait directement entre les deux terminaux) :

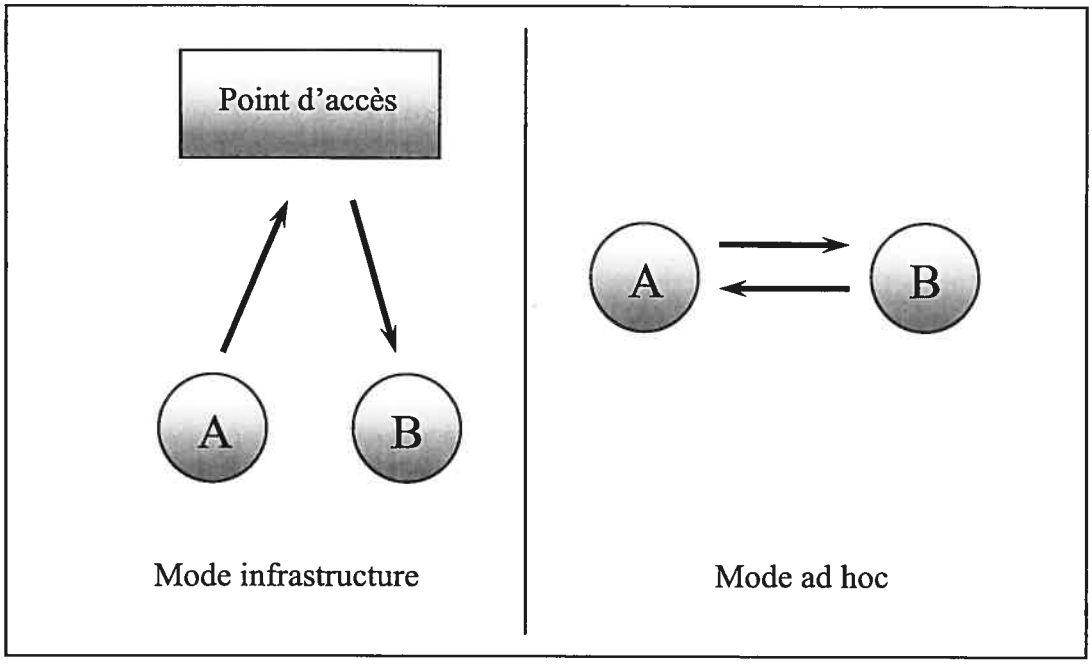

Figure 2.1. Communications en mode infrastructure et ad hoc.

2.2 Caractéristiques des réseaux ad hoc

Les réseaux ad hoc possèdent leurs propres caractéristiques :

- Autonomie (pas de routeurs dédiés) ;
- communication par liaison radio (bande passante limitée, taux d'erreurs élevé) ;
- topologie dynamique (mouvement des noeuds, mise en veille, ...) ;
- utilisation de batteries pour l'énergie (problème : capacité limitée) ;
- sécurité physique limitée (vol, endommagement, ...).

Attardons-nous plus en détail sur ces quelques points. L'autonomie des réseaux ad hoc tient au fait qu'ils ne bénéficient d'aucune infrastructure préexistante

ou d'appoint (station centrale, relais, ...). Un réseau ad hoc doit être facilement déployable, les nœuds pouvant joindre et quitter le réseau de façon totalement dynamique sans devoir en informer le réseau et si possible sans effet de bord sur les communications des autres membres (exemple : un hôte vient d'être informé de la tenue d'une conférence en ligne, il souhaite y assister). La nature ou l'origine des nœuds dans un réseau ad hoc importe peu (ordinateurs, PDA, téléphones mobiles, ...), chaque hôte devant simplement être équipé d'une interface de communication sans fil, qui peut elle aussi être différente d'un nœud à l'autre (Bluetooth, WiFi, UWB, ...). Deux nœuds étant à portée radio l'un de l'autre vont ainsi pouvoir rentrer en communication directement. Le schéma 2.2 suivant est un exemple de réseau ad hoc :

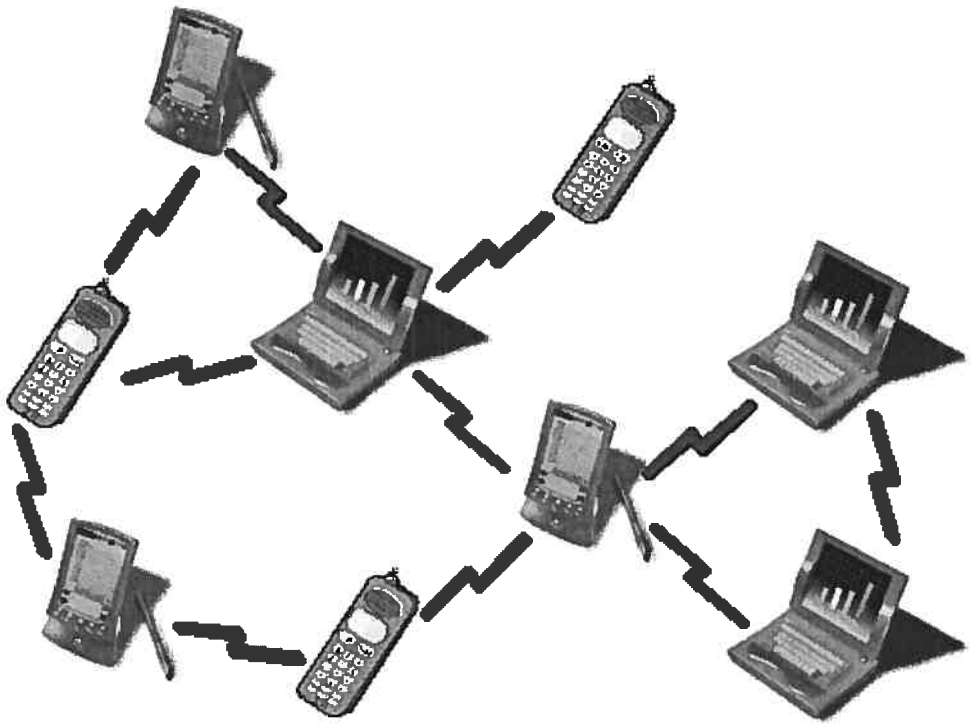

Figure 2.2. Exemple de réseau ad hoc.

On le sait bien, les liaisons sans fil ont une capacité inférieure à celles de leurs homologues câblées (la bande passante est limitée). De plus, le débit réel des

communications est variable et est souvent inférieur aux taux de transfert maximum de la radio. Tout ceci entraîne inexorablement une congestion dans le réseau lorsque les demandes d'utilisateurs augmentent (elles peuvent approcher ou dépasser la capacité du réseau). Cette demande ne cessera de croître avec le développement d'applications basées sur les réseaux mobiles.

Une autre caractéristique importante d'un réseau ad hoc est le caractère dynamique de sa topologie. En effet, à un instant donné, la position des nœuds, la configuration de leurs paramètres d'émission/réception ainsi que les niveaux de puissance de transmission et d'interférence entre les canaux forment un graphe multi saut aléatoire : le réseau ad hoc. Lorsque les nœuds bougent ou lors de l'ajustement de leurs paramètres d'émission/réception, il peut ainsi en résulter de rapides et imprévisibles changements dans la topologie du réseau. Si l'on ajoute à cela le fait que les communications aient lieu dans un environnement sans fil, le réseau pourrait être constitué de liaisons unidirectionnelles et bidirectionnelles. Les communications entre entités s'en trouvent de ce fait perturbées. On s'aperçoit donc que la difficulté de concevoir des protocoles de routage (unicast ou multicast) dans les réseaux ad hoc réside dans l'instabilité des chemins causée par la mobilité des nœuds. Les fréquents changements dans la topologie du réseau entraînent de fréquentes reconstructions de l'arbre et l'échange d'informations pour la mise à jour peut entraîner une congestion dans le réseau et la perte de paquets de données.

Il convient également de s'attarder sur la capacité d'énergie des nœuds d'un réseau ad hoc. En effet, la principale contrainte dans les communications sans fil est la durée de vie des terminaux mobiles dont le support énergétique représente souvent une batterie dont la capacité est limitée. Cette contrainte est beaucoup plus importante dans les réseaux ad hoc où les entités consomment leur propre énergie en routant des données pour d'autres entités. De plus, s'il arrive que des chemins soient plus empruntés que d'autres au sein du réseau, les nœuds qui les composent vont rapidement être dépourvus d'énergie et la durée de vie du réseau lui-même s'en

trouvera menacée. Pour ces nœuds, le plus important est donc de mettre en place des critères d'optimisation pour la conservation d'énergie.

Enfin, la sécurité physique des réseaux ad hoc est minimale. Outre le fait que l'environnement peut endommager les nœuds qui les composent (c'est le cas lorsque le réseau est déployé dans des zones à risque pour des applications spécifiques), on note que les réseaux sans fil sont généralement plus sensibles aux attaques que ne le sont les réseaux câblés (écoute passive, usurpation d'identité, déni de service, ...).

L'ensemble des caractéristiques que nous venons d'énoncer forme un ensemble de paramètres prédominants dont il faut tenir compte lors de la conception de nouveaux protocoles de communication. Il s'agit de développer des protocoles qui tiennent compte d'une part de la mobilité des nœuds, et par conséquent de la possible perte de liens, et d'autre part de la consommation d'énergie.

2.3 Applications des réseaux ad hoc

Les premières recherches sur les réseaux ad hoc ont débuté dans les années 60 au sein de la DARPA (*Defense Advanced Research Projects Agency*), organisme de recherche de l'armée américaine. L'armée visait le déploiement rapide de systèmes de communication dans des zones difficiles d'accès telles que les champs de bataille ou les lieux de catastrophes naturelles. Par la suite sont apparues des applications commerciales basées sur le recueil de données. Dans tous les cas, les applications des réseaux ad hoc sont celles qui nécessitent une rapidité de mise en place. De plus, de par leur fonctionnement autonome, les réseaux ad hoc sont idéals pour les environnements caractérisés par une absence (ou la non-fiabilité) d'une infrastructure préexistante. Une application quelque peu futuriste développée par la NASA vise à mettre en place un système de communications ad hoc pour des véhicules d'exploration de la planète Mars. Plus sérieusement, nous pouvons citer :

- Les applications militaires ;
- les applications commerciales ;
- les applications relatives à des situations d'urgence (incendies, tremblements de terre, ….) ou à des missions d'exploration (détection dans un terrain miné) ;
- les conférences et meetings ;
- la conduite coopérative entre véhicules ;
- et bien d'autres applications encore …

S'il s'agit de faire le lien avec le sujet qui nous intéresse, à savoir le routage multicast dans les réseaux ad hoc, on se rend compte de la complexité de définir de tels protocoles [15]. En effet, la densité du réseau, la vitesse de mobilité des nœuds, la fréquence de leur mouvement ainsi que leur capacité d'énergie (une voiture dispose d'une charge de batterie supérieure à celle d'un capteur par exemple) sont autant de facteurs qui peuvent varier d'une application à l'autre. Par exemple, certains protocoles se devront d'être plus performants dans des environnements ad hoc de densité faible (pour des conférences par exemple) alors que d'autres nécessiteront d'être extrêmement robuste à la mobilité (communications inter véhicules).

Chapitre 3

Protocoles multicast dans les réseaux ad hoc

L'inondation (ou *flooding*) est une façon d'effectuer du multicast dans les réseaux ad hoc. Les paquets de données sont propagés dans le réseau et chaque nœud se charge de continuer de les propager à ses voisins immédiats une et une seule fois. Il est évident que dans un environnement extrêmement dynamique, l'inondation peut s'avérer être une alternative fiable au multicast. Le ratio de livraison des paquets (le nombre de paquets de données réellement délivrés aux destinations par rapport au nombre de paquets de données censés être reçus) est proche de 100% [9, 10]. Cependant, il est important de constater qu'un grand nombre de paquets dupliqués sont ainsi envoyés. Cela amène à en tirer les conclusions suivantes : 1) la surcharge (ou *overhead*) peut augmenter d'une façon considérable ; 2) des collisions de paquets peuvent avoir lieu dans un réseau ad hoc à multiple accès. Il est donc nécessaire d'utiliser des protocoles spécialement orientés multicast pour les réseaux ad hoc.

Pour cela, il convient dans un premier temps de se demander si les techniques employées dans les réseaux filaires traditionnels peuvent être transposées aux réseaux ad hoc. Dans les réseaux filaires, la structure d'arbre (partagé ou non) est employée dans la majorité des protocoles multicast. C'est un concept parfaitement établi et mis en œuvre [4]. En effet, elle est le moyen le plus efficace en termes de ressources de connecter un certain nombre de nœuds et elle garantit la non-duplication des données. De plus, les décisions de routage sont simplifiées et consistent à propager les données vers les interfaces de sortie, exceptée celle par laquelle le message est arrivé. Vouloir transposer directement ces principes aux réseaux ad hoc peut s'avérer très inefficace. En effet, nous avons évoqué

précédemment les caractéristiques principales des réseaux ad hoc, parmi lesquelles les fréquents changements dans la topologie du réseau dus à la mobilité des nœuds. Pour cette raison, l'utilisation de protocoles multicast conçus spécifiquement pour les réseaux ad hoc se justifie totalement. Le multicast doit avant tout servir à réduire le nombre de ressources réseau employées et peut également être un moyen robuste de joindre des destinataires dont l'adresse est inconnue ou change régulièrement. Plusieurs protocoles multicast ont ainsi été proposés à ce jour mais aucun d'entre eux ne fait figure de standard.

Dans ce chapitre, nous allons tout d'abord faire un état de l'art des protocoles multicast proposés à ce jour pour les réseaux ad hoc. Nous allons présenter les différentes propositions et les regrouper par catégorie, basée sur la façon dont les routes sont créées vers les membres du groupe multicast. Les protocoles multicast pour les réseaux ad hoc peuvent être classés de cette façon :

- les protocoles employant une structure d'arbre ;
- les protocoles employant une structure de maillage ;
- les protocoles multicast explicites.

3.1 Protocoles employant une structure d'arbre

Les protocoles que nous allons présenter dans cette section ont été proposés avec comme intention d'étendre l'approche employant une structure d'arbre dans les réseaux filaires aux réseaux ad hoc. Nous allons donc détailler le principe et le fonctionnement de quelques uns des principaux protocoles multicast proposés à ce jour dans cette catégorie.

3.1.1 MAODV (Multicast Ad hoc On-demand Distance Vector) :

C'est le plus connu des protocoles multicast exploitant le principe d'arbre. MAODV (*Multicast Ad hoc On-demand Distance Vector*, [14]) est une extension du protocole unicast AODV [16]. Comme son nom l'indique, il crée les routes à la demande. Le mécanisme de découverte des routes est basé sur un cycle RREQ/RREP. Quand un nœud désire joindre un groupe multicast ou qu'il a des données à envoyer, il diffuse un paquet RREQ avec le flag JOIN et l'adresse de destination fixée à l'adresse du groupe multicast. Les nœuds intermédiaires qui reçoivent une requête RREQ d'adhésion à un groupe multicast dont ils ne sont pas membres continuent de propager le paquet RREQ. Seuls les membres du groupe multicast concerné sont autorisés à répondre à la requête par un message RREP. Au fur et à mesure de la propagation du message RREQ dans le réseau, les nœuds mettent en place des pointeurs afin d'établir la route inverse. Plus précisément, un nœud recevant un paquet RREQ d'adhésion à un groupe multicast (flag JOIN présent) met à jour sa table des routes en enregistrant l'information du prochain nœud (pour le retour vers le nœud désirant joindre le groupe). Cette route est par la suite utilisée afin de relayer une réponse à la requête. Lorsqu'un membre de l'arbre multicast avec une route vers la destination reçoit une requête RREQ, il y répond par un message RREP en unicast. Le message RREP est ainsi propagé jusqu'au nœud source désirant joindre le groupe. Il est à noter que lors de la propagation de la réponse RREP, les nœuds membres peuvent recevoir plusieurs messages RREP. Ceux-ci sélectionnent alors la meilleure route suivant certains critères et emploient un mécanisme basé sur un message d'activation multicast MACT (*Multicast Activation*) afin d'informer de la route sélectionnée.

Il arrive parfois qu'un nœud source ayant initié une requête RREQ reçoive plusieurs réponses RREP. Lorsque c'est le cas, il choisit une route, comme dans le cas précédent, en fonction de certains critères (numéro de séquence le plus petit, route la plus courte en nombre de sauts, ...). Lorsque la route est sélectionnée, il

envoie un message MACT en unicast à son prochain nœud. Celui-ci est utilisé pour activer le chemin vers le nœud qui a généré la réponse RREP. Ce processus est répété jusqu'à ce que le nœud qui a généré le message RREP soit atteint. La route est ainsi activée et opérationnelle. L'emploi d'un message d'activation permet d'une part de s'assurer que l'arbre multicast ne contient pas plusieurs chemins entre chaque nœud de l'arbre et d'autre part de permettre la propagation des messages de données uniquement le long de routes ayant été activées.

Si un nœud source ne reçoit de réponses RREP à sa requête pendant un certain temps, il inonde à nouveau le réseau avec un autre paquet RREQ. Après RREQ_RETRIES essais, il considère qu'il n'y a plus de membres dans l'arbre multicast qui peuvent être atteints et se déclare implicitement leader du groupe. Le leader du groupe est responsable de la diffusion périodique des messages HELLO au sein du group afin d'en maintenir la connectivité. Il est à noter que le premier membre du groupe multicast devient le leader de ce groupe. La désignation d'un leader permet au protocole de réagir rapidement aux changements de topologie dans l'arbre multicast. Pour ce faire, lorsqu'un nœud décide de ne plus faire partie du groupe multicast ou lorsqu'il y a détection d'un lien cassé, le nœud se situant le plus loin du leader (en aval de la cassure) prend la responsabilité de réparer le lien cassé et de relier les deux parties de l'arbre. Si l'opération échoue ou si elle est impossible, un nouveau leader est désigné pour le sous-arbre séparé par la cassure. Ce leader sera le nœud ayant initié la réparation s'il est un membre du groupe. Dans le cas contraire (s'il s'agit d'un nœud non membre), il se sépare du nouvel arbre en envoyant un message au prochain nœud et ce, jusqu'à ce qu'un nœud membre soit atteint. Celui-ci est désigné leader et sera en mesure de recevoir les messages HELLO propagés par le leader de l'arbre principal. A partir de ce moment là, la jointure entre les deux arbres peut avoir lieu.

La figure 3.1 suivante illustre la création de l'arbre et le mécanisme de découverte de routes dans le protocole MAODV. Le nœud source génère une requête de route RREQ et reçoit deux réponses RREP de la part de membres du groupe.

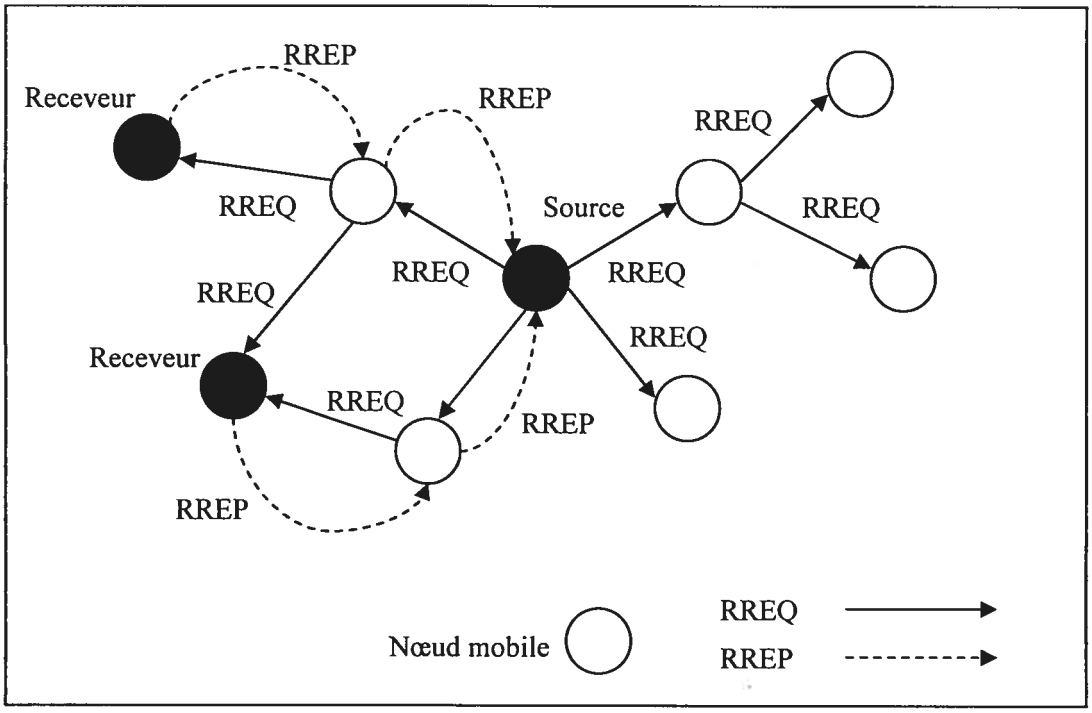

Figure 3.1. Propagation des messages RREQ et création de l'arbre multicast dans MAODV.

3.1.2 AMRIS (Ad hoc Multicast Routing protocol utilizing Increasing id-numberS) :

Un autre protocole proposé parmi les approches employant une structure d'arbre est AMRIS [17]. C'est un protocole « à la demande » (c'est-à-dire qu'il réagit lorsqu'un nœud fait une demande de route) qui construit un arbre multicast partagé afin de supporter plusieurs sources et plusieurs receveurs au sein d'une même session multicast. Le protocole assigne dynamiquement à chaque nœud de chaque session multicast un numéro ID que l'on désigne par msm-id (*multicast session member id*). L'ordre des msm-id est utilisé afin de diriger le flot des données multicast. Un arbre de livraison multicast dont la racine est le nœud ayant le plus petit msm-id est ainsi crée et le msm-id des nœuds est incrémenté au fur et à mesure que l'on s'éloigne de la racine (la racine est appelée Sid pour *Smallest-id*). Voyons plus en détail la création de cet arbre.

La première opération consiste en la sélection du Sid. Généralement, celui-ci est le nœud source ayant initié la session multicast. S'il y a plusieurs sources, le Sid est sélectionné parmi l'ensemble de ces sources. Une fois cette étape terminée, le Sid envoie un message NEW-SESSION à chacun de ses voisins. Ce message contient son propre msm-id ainsi que les métriques de routage. Lors de la réception de ce message, les nœuds voisins génèrent leurs propres msm-id en prenant soin que celui-ci soit plus grand que celui contenu dans le paquet. Lorsqu'un nœud reçoit plusieurs messages NEW-SESSION, il se base sur les métriques de routage afin de sélectionner le plus approprié. Le paquet NEW-SESSION continue d'être propagé avec le nouveau msm-id crée et ce, jusqu'à ce que tous les nœuds aient été atteints et que l'arbre soit crée. Ainsi, plus l'on s'éloigne du Sid, plus les msm-id augmentent.

Lorsqu'un nœud désire joindre le groupe multicast, il le fait par l'intermédiaire d'un message JOIN-REQ. Ce message est envoyé en unicast à un parent potentiel qui possède un msm-id plus petit que celui du nœud désirant joindre la session multicast. Si le nœud parent est un membre du groupe, il répond à la requête de jointure par un message JOIN-ACK. Dans le cas contraire, il contacte son parent en lui faisant suivre le message JOIN-ACK, jusqu'à atteindre un membre du groupe multicast. Si un nœud ne reçoit pas de réponse à sa requête ou s'il reçoit une réponse négative (impossibilité de joindre un parent potentiel), il doit initier une procédure *Branch Reconstruction* afin de joindre l'arbre. Elle consiste à diffuser un message JOIN-REQ dans le réseau en utilisant une technique de recherche par anneaux croissants. Cette technique recherche des nœuds dans toutes les directions situées à une portée (TTL ou *Time-To-Live*) limitée. Le TTL est incrémenté au fur et à mesure en cas d'échec, d'où la notion d'anneaux croissants. Les nœuds qui reçoivent le message et qui font partie du groupe multicast doivent l'acquitter afin d'informer le nœud de son appartenance au groupe.

En cas de perte de lien, le nœud avec le plus grand msm-id se charge de rejoindre l'arbre soit en envoyant une demande d'adhésion à un parent potentiel (un nœud ayant un msm-id plus petit) soit en lançant la procédure *Branch*

Reconstruction. AMRIS est indépendant de tout protocole de routage unicast et utilise ses propres messages d'avertissement périodiques afin de maintenir sa table de voisinage. La déconnection des liens est ainsi détectée par l'intermédiaire d'un mécanisme de messages à balise (ou messages *beacon*) : un nœud qui n'en reçoit pas pendant un certain laps de temps considère que son ou ses voisins se sont déplacés et sont sortis de son champ d'écoute. Ainsi, si un lien se casse, des paquets de données peuvent être perdus tant que le lien n'est pas réparé.

3.1.3 AMRoute (Ad hoc Multicast Routing) :

Enfin, parmi les protocoles basés sur une structure d'arbre, il convient de mentionner AMRoute [18]. Ce protocole crée un arbre multicast partagé bidirectionnel en utilisant des tunnels unicast qui servent de lien entre les nœuds membres. Ces tunnels permettent la diffusion des données, qui se fait donc uniquement entre les émetteurs et les récepteurs du groupe multicast. Les nœuds qui ne sont pas membres ne relaient pas les paquets de données et ne nécessitent pas le support d'un protocole multicast.

Certains nœuds de l'arbre jouent le rôle de noyau logique (*core*). Il y en a au moins un par groupe et ils sont responsables de la maintenance de l'arbre, de la détection de nouveaux membres et de la mise en place des tunnels de l'arbre. Il est à noter que les noyaux logiques ne sont pas des points fixes et qu'ils ne représentent donc pas un point faible de l'architecture. Initialement, chaque membre du groupe se déclare noyau pour son propre groupe (de taille 1) et diffuse périodiquement des messages JOIN-REQ afin de découvrir des segments de maillage disjoints pour le groupe. Lorsqu'un membre reçoit un message JOIN-REQ de la part d'un noyau du même groupe mais d'un segment de maillage différent, il y répond par un JOIN-ACK et marque ce nœud noyau comme un voisin de maillage. Le nœud qui reçoit le message JOIN-ACK en fait de même. Après la création du maillage s'en suit la

procédure d'élection du noyau au sein du maillage. En effet, il se peut qu'il y ait deux noyaux dans le même maillage lors de la fusion. Cette procédure consiste simplement en un algorithme déterministe qui choisit un unique noyau par maillage. Suite à cela, le noyau élu transmet périodiquement des paquets TREE-CREATE dirigés vers chacun de ses voisins de maillage. Cela a pour but de créer l'arbre partagé. Lorsqu'un nœud reçoit un message TREE-CREATE non dupliqué de la part de l'un de ses voisins de maillage, il continue de le propager vers ses propres voisins. Si un message TREE-CREATE dupliqué est reçu, un message TREE-CREATE-NAK est renvoyé le long du chemin dont est issu le message TREE-CREATE. Le nœud recevant le paquet TREE-CREATE-NAK en déduit que le lien ne fait pas partie de l'arbre (mais il fait toujours partie du maillage). Ce processus aboutit à la création de l'arbre partagé à partir du maillage. Lorsqu'un nœud désire quitter le groupe multicast, il envoie un message JOIN-NAK à ses voisins et ne propage pas les paquets de données qu'il reçoit.

La clé du bon fonctionnement d'AMRoute est l'utilisation de liens de maillage virtuels afin d'établir la structure d'arbre (AMRoute fait partie de ce que l'on appelle les protocoles hybrides mais se classe dans la catégorie des protocoles employant une structure d'arbre). Ainsi, cette structure peut rester identique même en cas de changement de topologie car il existe toujours des routes dans le maillage entre les membres du groupe. Cependant, les inconvénients de ce protocole sont qu'il nécessite l'emploi d'un protocole de routage unicast afin de maintenir la connectivité du groupe. De plus, il souffre parfois de la création de boucles temporaires et de la création d'arbres non optimaux lorsque la mobilité est élevée.

La figure 3.2 suivante montre la formation de l'arbre AMRoute. Les nœuds A et B joignent simultanément le groupe multicast, se désignent chacun noyau logique de leur propre groupe et commencent à transmettre des messages JOIN-REQ avec un TTL limité. La procédure d'élection du noyau a lieu lorsque l'un des deux nœuds reçoit le message de l'autre. Imaginons que B perd. A est donc noyau logique et il existe un lien dans l'arbre connectant les nœuds A et B (un tunnel). Supposons que le

nœud C veut maintenant se joindre au groupe. Il se désigne noyau logique de son propre groupe de taille 1 et transmet des messages JOIN-REQ. Le nœud B étant le plus près de C, celui-ci va recevoir le message en premier. A partir de ce moment là, il existe un lien de maillage entre B et C. La procédure de sélection du noyau au nœud B va élire le nœud C. Le nœud B va donc propager les messages TREE-CREATE de la part de C vers A et ce dernier va également déduire que C est le noyau logique du nouveau groupe. A n'est plus maintenant un noyau logique et le lien de maillage entre B et C devient un lien d'arbre.

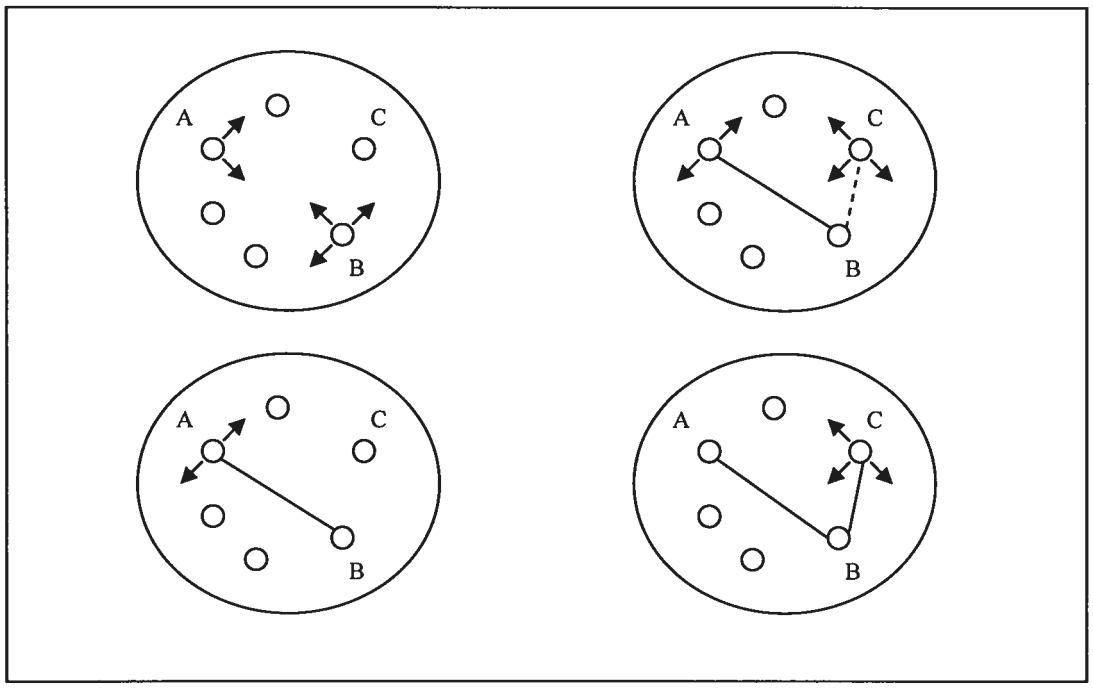

Figure 3.2. Formation de l'arbre à partir du maillage dans AMRoute.

3.2 Protocoles employant une structure de maillage

De nombreuses études [19] ont montré que les protocoles multicast employant une structure d'arbre n'étaient pas nécessairement bien adaptés aux réseaux ad hoc. En effet, on peut leur reprocher la fragilité de la structure due au fait

qu'il n'existe qu'un seul chemin pour relayer les données. De plus, les fréquents changements dans la topologie du réseau, dans un environnement fortement mobile par exemple, vont à coup sûr entraîner de fréquentes reconfigurations de l'arbre (dès qu'un lien se casse). Pour palier à ce problème, certains travaux ont proposé de maintenir une structure maillée (*mesh*) qui est plus robuste car redondante (plusieurs chemins alternatifs existent). Nous allons détailler dans cette section deux protocoles utilisant cette approche.

3.2.1 ODMRP (On Demand Multicast Routing Protocol) :

Comme MAODV, ODMRP [13] est un protocole réactif : il établit les routes à la demande. Cependant, il se distingue de MAODV en employant une structure de maillage ; c'est-à-dire qu'un sous-ensemble de nœuds (que l'on appelle aussi *Forwarding Group* ou FG) sont choisis pour relayer les paquets de données multicast. Les nœuds qui font partie du FG sont maintenus par une inondation périodique de messages de contrôle. Ils doivent garantir l'existence d'au moins un chemin entre chaque source et chaque récepteur du groupe.

ODMRP fonctionne suivant une phase requête/réponse. Quand une source a des données à envoyer mais pas de route ni d'informations d'appartenance à un groupe multicast, elle diffuse périodiquement un message JOIN_QUERY. Lorsqu'un nœud voisin reçoit un paquet JOIN_QUERY non dupliqué, il mémorise dans sa table de routage unicast le numéro ID du nœud en amont. Cette information permet de connaître le chemin unicast inverse pour la réponse. Puis il continue de propager le message. Quand le paquet JOIN_QUERY atteint un receveur du groupe multicast, celui-ci génère un paquet JOIN_REPLY (qui est en fait une table contenant les prochains nœuds pour le chemin inverse) et le diffuse à ses voisins. Lors de la réception d'un paquet JOIN_REPLY, un nœud vérifie si l'adresse du prochain nœud d'une des entrées de la table correspond avec sa propre adresse et si c'est le cas, il

réalise qu'il est sur le chemin vers la source : il va donc faire partie du FG pour cette source et l'indique par l'intermédiaire d'un flag spécial. Puis il continue de diffuser le paquet JOIN_REPLY jusqu'à atteindre la source via le chemin le plus court. Ce processus établit ou met à jour les routes entre sources et receveurs et construit le FG. Les informations de route et d'appartenance au groupe sont périodiquement mises à jour par l'intermédiaire de messages JOIN_QUERY. Lorsqu'un nœud désire quitter le groupe, il arrête simplement d'envoyer des messages JOIN_QUERY. S'il désire ne plus recevoir de données de la part d'un groupe particulier, il ne répond pas aux requêtes pour ce groupe. On remarque donc qu'aucun paquet de contrôle n'est nécessaire pour les opérations de jointure et de départ dans ODMRP.

A la différence des protocoles basés sur une structure d'arbre, il va donc y avoir plusieurs routes entre deux nœuds, d'où la structure de maillage. La présence de plusieurs chemins (lorsqu'ils existent) entre deux nœuds permet de continuer à délivrer les données même lorsque le chemin principal devient impraticable. Il est à noter que seuls les nœuds faisant partie du FG ou les membres du groupe multicast sont aptes à relayer les paquets de données. OMDRP se révèle donc être plus robuste à la cassure de liens ou à la perte de nœuds. La figure 3.3 illustre la création du maillage dans ODMRP. La source diffuse périodiquement un message JOIN_QUERY. Deux receveurs du groupe multicast y répondent par l'intermédiaire de messages JOIN_REPLY, ce qui permet de construire ou de mettre à jour le FG.

Récemment, ODMRP a été adapté pour tenir compte du mouvement des nœuds en utilisant un modèle de prédiction de mobilité [20]. Pour cela, les réseaux doivent disposer d'un système GPS (*Global Positioning System*) afin de collecter les informations de localisation et de mobilité. L'expiration d'une route peut être estimée et les receveurs peuvent ainsi sélectionner le chemin qui restera valide le plus longtemps. De plus, les sources peuvent reconstruire les routes en prévision d'une future cassure de liens. Le protocole n'en devient que plus robuste à la mobilité mais cela a un inconvénient : l'utilisation du GPS entraîne un coût plus élevé en terme d'énergie et d'argent.

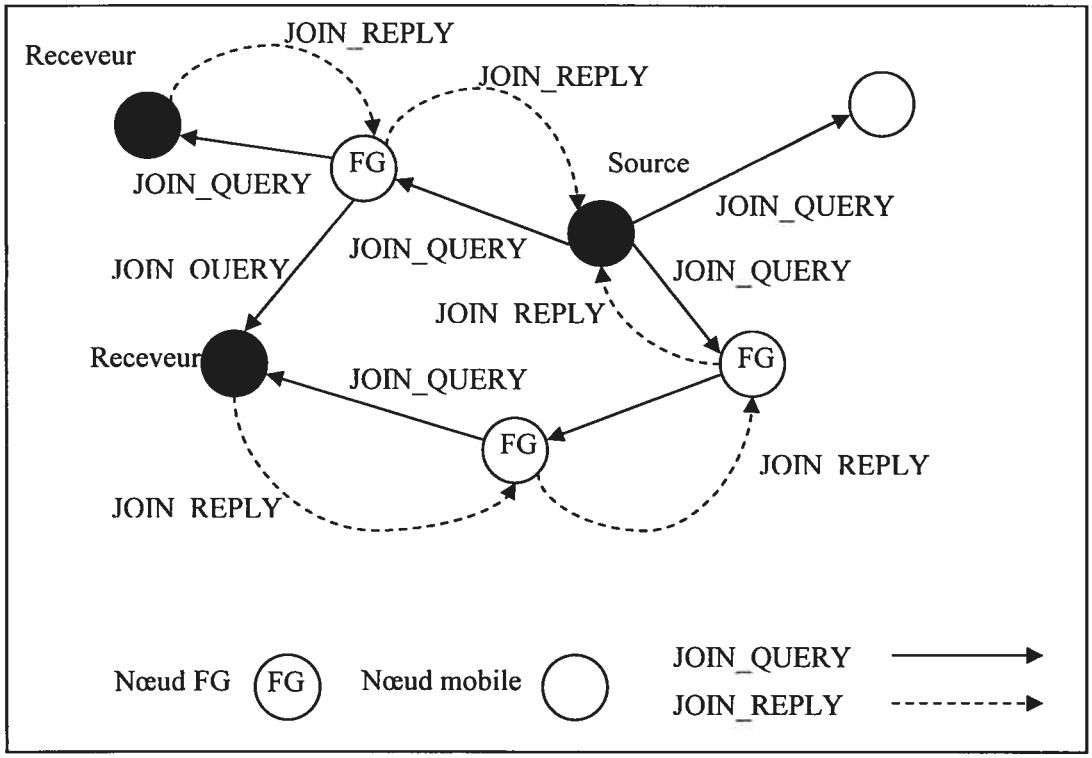

Figure 3.3. Formation du maillage multicast dans OMDRP.

3.2.2 CAMP (Core-Assisted Mesh Protocol) :

Un autre protocole intéressant parmi ceux employant une structure maillée est CAMP [21, 22]. C'est un protocole de routage multicast qui construit une structure de maillage partagée pour chaque groupe multicast. Dans CAMP, un ou plusieurs nœuds jouent le rôle de noyau (ou *core*), prenant ainsi en charge les opérations d'adhésion au groupe. Ils sont utilisés pour limiter le trafic de contrôle nécessaire à la jointure d'un nœud au groupe multicast. Les opérations d'inondation du réseau sont de ce fait inutiles.

Lorsqu'un nœud désire joindre la session multicast, il vérifie dans un premier temps s'il y a des membres du groupe parmi ses voisins. Si c'est le cas, il les informe de sa nouvelle appartenance au groupe par l'intermédiaire d'un message de mise à

jour CAMP_UPDATE. Sinon, deux possibilités s'offrent à lui ; il peut soit 1) essayer de joindre un membre du groupe par l'intermédiaire d'un mécanisme de recherche par anneaux croissants ; soit 2) contacter un des noyaux du groupe multicast. S'il opte pour la première solution, n'importe quel nœud membre peut répondre par un message JOIN_ACK qui est propagé jusqu'à l'initiateur de la requête. Si la deuxième solution est choisie, le chemin pour joindre le noyau va alors être incorporé entièrement au maillage. Un nœud receveur détermine périodiquement s'il reçoit des paquets de données de la part des voisins qui sont sur le chemin inverse le plus court vers la source. Si ce n'est pas le cas, le nœud envoie un message HEART_BEAT le long du chemin inverse le plus court. Ce processus permet de s'assurer que tous les chemins inverses entre les sources et les receveurs sont bien inclus dans le maillage.

CAMP a l'avantage d'une part de ne pas utiliser l'inondation et d'autre part que les requêtes soient propagées uniquement en direction des membres du maillage. Cependant, il repose sur un protocole unicast sous-jacent pour garantir des distances correctes vers chaque destination en un temps fini.

3.3 Protocoles multicast explicites

Les deux approches que nous venons de présenter (structure d'arbre et structure de maillage) ont un inconvénient majeur : elles sont majoritairement adaptées aux réseaux dans lesquels il y a un nombre limité de groupes multicast de grande taille. Ceci est particulièrement bien approprié lorsqu'il s'agit de distribuer du contenu à une grande quantité de receveurs mais il peut y avoir des problèmes d'extensibilité lorsqu'il s'agit de traiter un grand nombre de petits groupes (dans des applications telles que la conférence). De plus, créer et maintenir à jour l'arbre/le maillage entraîne indéniablement une augmentation considérable de la surcharge. Cela est d'autant plus préjudiciable dans les réseaux ad hoc où les nœuds se

déplacent fréquemment et aléatoirement. Pour pallier ce problème, le schéma Xcast a été proposé.

Le principe général est que la source encode explicitement la liste de toutes les destinations dans l'en-tête du paquet. Cette famille de protocoles se focalise sur les petits groupes multicast (SGM pour *Small Group Multicast*) et suppose qu'un protocole de routage unicast sous-jacent s'occupe de délivrer les paquets aux destinations en se basant sur les adresses contenues dans l'en-tête de celui-ci. Ainsi, aucun protocole de routage multicast n'est utilisé et cela a l'avantage que les paquets prennent automatiquement le chemin approprié déterminé par la table de routage unicast. Les étapes suivies par un routeur qui reçoit un paquet Xcast sont les suivantes :

- Il examine l'en-tête des paquets ;
- il partitionne les adresses de destination en se basant sur l'information du nœud prochain ;
- il fait suivre le paquet avec l'en-tête Xcast appropriée à chacun des nœuds.

La figure 3.4 suivante illustre le fonctionnement classique du schéma Xcast. Le nœud S est la source de la session multicast et les nœuds R1 à R6 sont les receveurs. Lorsque la source envoie un paquet de données, l'en-tête est {N1 : R1, R2, R3, R4, R5, R6} où N1 est le prochain saut. Lors de la réception du paquet, le nœud N1 détermine que le nœud N2 est le prochain saut pour l'ensemble des destinations contenues dans l'en-tête. Il modifie donc cet en-tête et fait suivre le paquet. Lorsque N2 le reçoit, il s'aperçoit que le prochain saut pour les nœuds R1 et R2 est N3 et celui pour les nœuds R3 à R6 est N4. L'en-tête devient {N3 : R1, R2 ; N4 : R3, R4, R5, R6}. Et ainsi de suite jusqu'à ce que tous les receveurs aient reçu le paquet.

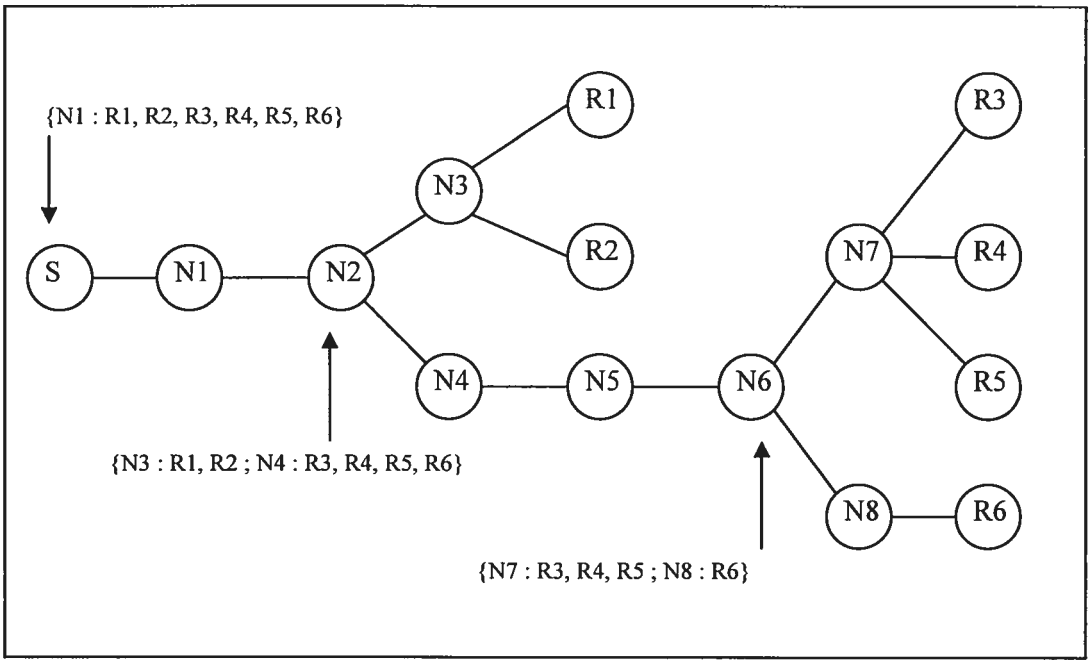

Figure 3.4. Cheminement d'un paquet de données dans le schéma Xcast.

Plusieurs protocoles ont étendu le principe du schéma Xcast [23, 24]. Nous allons détailler dans cette section quelques uns d'entre eux, présenter leurs avantages et inconvénients et mettre en avant la nécessité d'un autre protocole étendant Xcast pour les petits groupes multicast.

3.3.1 DDM (Differential Destination Multicast) :

DDM [7] est l'un des premiers protocoles à étendre le schéma Xcast. Dans ce protocole, c'est la source qui s'occupe de contrôler l'appartenance au groupe multicast. Elle utilise un en-tête DDM spécial et encode dans l'en-tête de chaque paquet à la fois la liste des destinations mais également le prochain saut pour atteindre chacune d'elles. Il est à noter que DDM peut fonctionner selon deux modes. Le premier est le mode sans état (ou *stateless*) dans lequel les nœuds intermédiaires ne maintiennent pas d'état pour chacune des sessions multicast. Lors de la réception

d'un paquet, ils extraient simplement les adresses contenues dans l'en-tête du paquet et en déduisent le chemin à suivre par le paquet pour atteindre les destinations restantes. C'est le protocole de routage unicast sous-jacent qui s'occupe de délivrer les paquets. Le second mode d'utilisation de DDM est le mode dit *soft state* : chaque nœud le long du chemin mémorise les destinations auxquelles le paquet a été propagé la fois précédente ainsi que le prochain saut pour atteindre chacune des ces destinations. Ce mécanisme de cache permet de ne pas lister la totalité des destinations dans l'en-tête des paquets. Si un changement a lieu au niveau routage unicast, un nœud devra simplement informer ses nœuds avals des différences qui se sont produites depuis la dernière réception d'un paquet, d'où le nom du protocole (*Differential Destination Multicast*).

Lorsqu'un nœud désire joindre la session multicast, il envoie un message JOIN en unicast vers la source. Celle-ci y répond par un message ACK et ajoute le nœud à sa table des receveurs. Les receveurs doivent régulièrement envoyer un message JOIN en direction de la source pour prévenir de leur volonté de toujours faire partie du groupe multicast. De la même manière, un nœud désirant se séparer du groupe le fait par l'intermédiaire d'un message LEAVE envoyé en unicast vers la source. On remarque que c'est la source qui s'occupe des opérations d'appartenance au groupe. Les paquets de données sont envoyés par la source après avoir préalablement encodé la liste complète des prochains sauts et les destinations correspondantes dans l'en-tête du paquet (en se basant sur la table des receveurs). Puis elle diffuse le paquet à l'ensemble de ses voisins. Les nœuds qui reçoivent ce paquet partitionnent l'en-tête afin de vérifier s'ils font partie de la liste des prochains sauts, et ainsi de suite.

Conformément au schéma classique Xcast, il est facile de remarquer que dans le protocole DDM, plus le nombre de receveurs sera grand et plus l'en-tête des paquets de données va augmenter en conséquence. Cette importante surcharge peut résulter en une perte de temps et d'énergie lorsque les nœuds réceptionnent un paquet (extraction de l'information, décision de routage puis propagation de l'information).

De plus, le fait que les opérations d'appartenance au groupe multicast soient concentrées à la source peut engendrer ce que l'on appelle le problème de JOIN_IMPLOSION. Imaginons en effet que les membres de la session multicast envoient périodiquement des messages à la source (comme c'est le cas dans DDM pour maintenir l'appartenance au groupe), la source ne pourra traiter toutes les demandes ou le temps pour y répondre sera important et ce, même pour une session multicast de petite taille. Pour cela, le protocole E²M a été introduit et se propose de diminuer la surcharge engendrée et d'offrir un certain degré d'extensibilité relativement au nombre de membres du groupe multicast.

3.3.2 E²M (Extended Explicit Multicast) :

E²M [8] a pour but de surmonter les limitations du schéma Xcast. Pour cela, il introduit le principe de *Xcast Forwarder* (XF) : certains nœuds sont dynamiquement sélectionnés suivant le nombre de nœuds avals qu'ils desservent. Ce concept a déjà été mis en œuvre dans les réseaux filaires lorsqu'un routeur de bord (ou DR pour *Designated Router*) rejoignait la session après avoir été préalablement informé par un de ces nœuds en aval qu'il souhaitait se joindre au groupe. Dans ce cas, si le routeur dessert plus d'un membre, la source n'aura besoin de mentionner que l'adresse du routeur. Si l'on applique ce principe aux réseaux ad hoc, cela peut engendrer quelques problèmes. En effet, la source n'ayant aucune connaissance des nœuds desservis par le routeur, lorsque celui-ci bouge (phénomène très fréquent dans MANETs), les nœuds qu'il dessert ne recevront pas de données durant le laps de temps nécessaire à sa réintégration dans le groupe. Il est donc très difficile de sélectionner correctement les routeurs de bord. Voyons la technique utilisée dans E²M pour sélectionner les nœuds XF.

Un nœud décide de devenir XF lorsque le nombre de nœuds qu'il dessert est supérieur à une certaine valeur et qu'il est un nœud de branchement. Par exemple,

cette valeur peut être fixée à 8 pour IPv4 afin d'éviter de multiples transmissions du même paquet. Lorsqu'un noeud décide de devenir XF, il envoie un paquet XF_JOIN à la source (qui contient l'ensemble des nœuds avals qu'il dessert) et celle-ci lui répond par un paquet XF_ACK en unicast. La source ajoute le nœud XF à sa table des XF ainsi que l'ensemble des destinations dont il est le responsable. À partir de ce moment, la source n'encodera plus que l'adresse du nœud XF dans l'en-tête du paquet au lieu d'y inclure l'ensemble des destinations. On voit donc que E²M minimise le problème d'implosion de messages MEMBER_JOIN en envoyant un seul paquet XF_JOIN pour tous les membres. Cependant, si le nombre de nœuds souhaitant intégrer la session multicast est faible, ils se contenteront d'envoyer un message MEMBER_JOIN directement à la source et E²M fonctionnera donc comme le schéma classique Xcast en encodant la liste des destinations dans l'en-tête. Périodiquement, les nœuds desservis par un XF lui envoient des messages MEMBER_REFRESH pour lui faire savoir qu'ils y sont toujours rattachés. Par la suite, les nœuds XF envoient des messages XF_REFRESH à la source pour l'informer de changements potentiels dans la liste des nœuds qu'ils desservent.

Lors du déplacement d'un nœud XF, la source en est informée lors de la génération d'un paquet d'erreur de route AODV ou DSR suivant le protocole de routage unicast utilisé. Dès lors, elle obtient la liste des destinations servies par cet XF, les encode explicitement dans l'en-tête des futurs paquets de données et supprime le nœud XF de sa table des XF. Lorsqu'un nœud quelconque bouge, le nœud XF qui le dessert ne recevra pas ses messages MEMBER_REFRESH périodiques. La source sera alors informée du déplacement du nœud lors de la transmission du prochain XF_REFRESH par le nœud XF (pour notifier les changements potentiels).

La figure 3.5 suivante illustre le principe de XF. Le nœud S est la source et les nœuds R1 à R6 sont les receveurs. Le nœud N6 a décidé de devenir XF : ainsi, l'en-tête du paquet partant de S est {N1 : R1, R2, N6} au lieu de {N1 : R1, R2, R3, R4, R5, R6} dans des protocoles comme DDM par exemple :

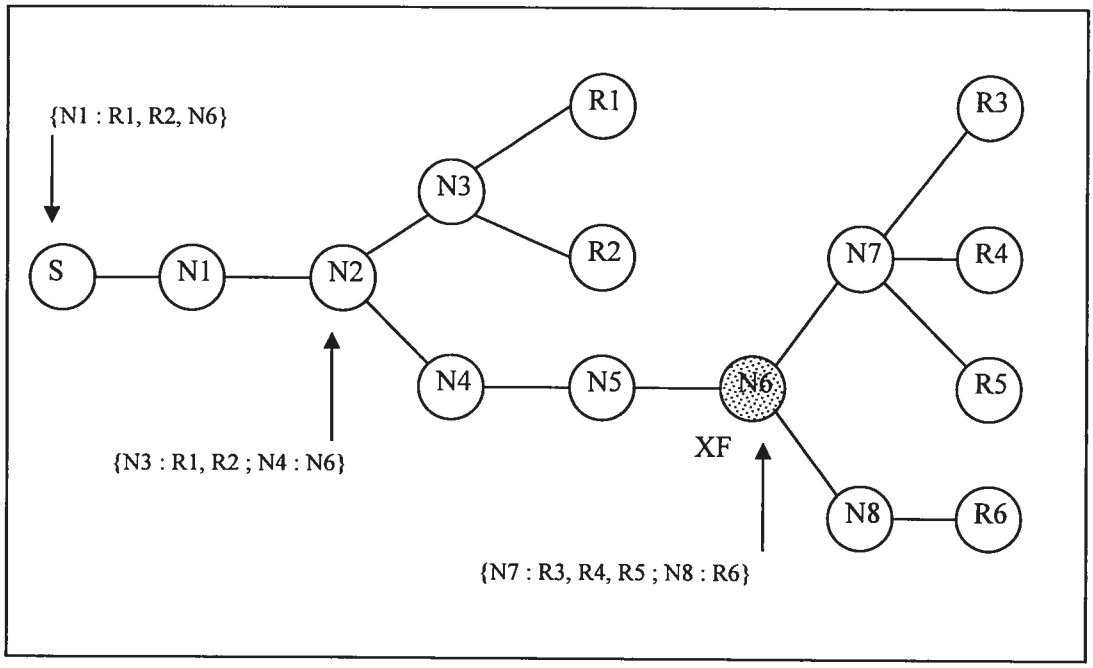

Figure 3.5. Cheminement d'un paquet de données dans E²M.

3.4 Conclusion

Quelle est la meilleure configuration à choisir ? Les chemins doivent-ils être crées à la demande ou déterminés une fois et mis à jour périodiquement ? Les paquets de contrôle sont-ils diffusés sur l'ensemble du réseau ou limités aux nœuds dans l'arbre multicast ? Ce sont autant de questions essentielles lors de la définition d'un protocole multicast pour les réseaux ad hoc. De nombreuses études ont été menées afin de comparer les protocoles décrits dans ce chapitre [9, 25]. Nous allons vous présenter brièvement les résultats qui ressortent de ces études en commençant par comparer chacun d'eux dans leurs catégories respectives. Le tableau 3.1 illustre les caractéristiques principales des protocoles multicast que nous venons de présenter.

Protocoles	Topologie multicast	Sans boucle	Dépendance à un protocole unicast	Messages Périodiques	Diffusion de paquets de contrôle
MAODV	Arbre	Oui	Oui	Oui	Oui
AMRIS	Arbre	Oui	Non	Oui	Oui
AMRoute	Arbre/Maillage	Non	Oui	Oui	Oui
ODMRP	Maillage	Oui	Non	Oui	Oui
CAMP	Maillage	Oui	Oui	Oui	Non
DDM	Arbre	Oui	Oui	Oui	Non
E²M	Arbre	Oui	Oui	Oui	Non
Flooding	Maillage	Oui	Non	Non	Non

Tableau 3.1. Comparaison des protocoles multicast dans les réseaux ad hoc.

Parmi les protocoles employant une structure d'arbre, AMRoute se comporte assez bien lorsque la mobilité est nulle (le ratio de livraison des paquets est proche de 100%) mais ses performances décroissent rapidement lorsque la vitesse augmente. De plus, la création de boucles temporaires lors de la phase de reconstruction de l'arbre et la possibilité de formation d'un arbre non optimal entraînent de sérieuses congestions et des pertes de paquets conséquentes. Tout ceci, ajouté au fait qu'il est dépendant d'un protocole unicast sous-jacent, font d'AMRoute un protocole bien en dessous des performances de ses concurrents. AMRIS est également performant lorsque la mobilité est faible et la charge du réseau également. Cependant, tout comme AMRoute, il est très sensible à la vitesse de mobilité des nœuds. La présence d'un seul chemin entre les membres du groupe peut entraîner des pertes de paquets lorsqu'il y a cassure. Le mécanisme de détection des cassures (par l'intermédiaire de messages à balise) aide à contourner ce problème. Mais cette méthode n'est pas tout à fait optimale. Pour le vérifier, plaçons nous dans le contexte suivant : les messages à balise sont envoyés toutes les secondes et il est nécessaire d'attendre trois échecs consécutifs pour lancer la procédure de reconstruction. Il faudra alors attendre 3 secondes au minimum pour que la reconstruction ait lieu et des paquets pourraient

être perdus pendant cet intervalle de temps. D'un autre côté, si l'on envoie des messages à balise plus souvent, cela peut augmenter la collision de paquets. On voit bien que l'intervalle de transmission des messages à balise est à choisir avec précaution et les recherches pour améliorer AMRIS portent actuellement sur ce domaine. Au final, MAODV se révèle être le plus performant des protocoles multicast employant une structure d'arbre. Les routes sont construites à la demande en un temps restreint et les cassures de liens sont réparées efficacement. De plus, il ne souffre pas de la création de boucles temporaires. Cependant, comme tous les autres protocoles de cette catégorie, il reste sensible au déplacement des nœuds, caractéristique essentielle des réseaux ad hoc. En effet, ces protocoles, bien qu'étant économiques en terme de bande passante, ne sont pas robustes à la mobilité. Au contraire, la présence de chemins alternatifs dans les protocoles employant une structure de maillage leur assure de ne pas souffrir de la perte d'un lien ou de l'éloignement d'un nœud.

Voyons donc maintenant les protocoles qui utilisent une structure maillée pour délivrer les paquets aux destinations. Les études montrent que CAMP offre globalement de meilleures performances que les protocoles à base d'arbre. Il n'est cependant pas aussi robuste qu'espéré lors de l'introduction de la mobilité. Ceci est du au fait que dans CAMP, les destinations les plus éloignées comportent moins de chemins redondants que celles au centre du maillage. Il y a donc une plus grande probabilité de cassure aux alentours de ces nœuds et par conséquent, des nœuds ancres peuvent ne pas recevoir les paquets qui leur sont destinés. Il convient donc de sortir du lot ODMRP. En effet, bien que n'ayant pas un ratio de livraison de paquets aussi élevé que l'inondation (pour lequel nous avons montré qu'il n'était pas efficace pour le multicast dans les réseaux ad hoc), il est très robuste à la mobilité et ce, même dans des environnements extrêmement dynamiques. La structure de maillage offre une redondance de routes, ce qui augmente les chances qu'un paquet arrive à destination. L'inconvénient du protocole ODMRP est qu'il est orienté source (on dit aussi qu'il est *sender-initiated*). En effet, il peut y avoir une consommation excessive de paquets de contrôle lorsque le nombre de sources augmente. Malgré tout,

ODMRP se révèle être à ce jour le protocole multicast le plus efficace pour les réseaux ad hoc lorsque l'environnement dans lequel il est utilisé est un environnement peuplé d'un petit nombre de larges groupes multicast.

En effet, intéressons-nous maintenant aux protocoles multicast explicites, spécialement conçus pour les petits groupes. Nous avons discuté auparavant de la nécessité de protocoles adaptés à ce genre de configuration. Peu de comparaisons ont été menées sur les deux protocoles que nous avons présentés, DDM et E²M. Néanmoins, si l'on se réfère à une métrique telle que la taille moyenne de l'en-tête des paquets de données, il est à noter que E²M se comporte mieux que DDM lorsque la taille du groupe augmente : ceci est du au fait que E²M sélectionne un nœud XF pour se charger d'un sous-ensemble de nœuds. De plus, la sélection de cet XF permet à E²M de n'envoyer qu'un seul paquet de contrôle pour tous les nœuds dont il a la charge (le problème de JOIN_IMPLOSION est ainsi réglé). Bien entendu, pour les deux protocoles, plus la taille du groupe augmente et plus la taille de l'en-tête des paquets de données augmente (puisque chaque destination est encodée dans l'en-tête, avec des variantes). À première vue, il convient donc de dire qu'E²M est plus performant que DDM. Cependant, E²M a un sérieux inconvénient : la gestion de la mobilité des nœuds est basique. C'est la source qui se charge de traiter ce problème lorsqu'elle en est informée. Il n'y a pas, dans E²M, de mécanisme de réparation de l'arbre, ce qui peut engendrer des pertes de paquets. Qui plus est, un nœud XF va consommer plus d'énergie que n'importe quel autre nœud dans le réseau afin de gérer l'encapsulation, la duplication et la transmission des paquets de données aux nœuds qu'il dessert. Nous venons de mettre en avant deux points clés qui justifient l'utilisation d'un autre protocole multicast pour les réseaux ad hoc : la gestion de la mobilité et la consommation d'énergie des nœuds.

Dans le chapitre suivant, nous allons présenter une solution apportée à ces problèmes. Le protocole EM2NET est un protocole multicast explicite pour les petits groupes qui étend E²M et propose en plus un mécanisme de réparation de l'arbre. Il permet un balancement de la charge entre les nœuds de branchement et offre une

meilleure conservation d'énergie. À l'aide de simulations avancées, nous allons comparer ce protocole avec E²M et nous allons surtout montrer en quoi les protocoles explicites sont particulièrement bien adaptés aux petits groupes multicast en comparant EM2NET avec ODMRP et MAODV. En effet, aucune étude n'a à ce jour été menée pour montrer l'influence de la densité du réseau sur les performances des protocoles.

Chapitre 4

Analyse et simulations

4.1 EM2NET : un protocole Xcast pour MANETs

EM2NET [11] est un protocole multicast explicite pour les réseaux ad hoc introduit pour les petits groupes. Il utilise les services unicast et construit un arbre multicast qui consiste en des nœuds particuliers appelés nœuds IN (*Intercepting Nodes*). Un nœud IN est un membre du groupe déjà dans l'arbre ou un nœud de branchement. À la différence de E²M, EM2NET construit l'arbre des plus courts chemins entre une source et les destinations. Le problème du routage asymétrique est ainsi réglé (le chemin entre A et B peut différer du chemin entre B et A à cause de la portée de transmission des nœuds dans un réseau ad hoc). De plus, il tient compte de la mobilité des nœuds en offrant un mécanisme de reconstruction de l'arbre et permet également un balancement de la charge entre les nœuds IN, ce qui assure une meilleure conservation d'énergie. Voyons plus en détail la conception du protocole EM2NET.

4.1.1 Messages et structures de données

EM2NET utilise deux types de paquets : les paquets de données et les paquets de contrôle. Chaque nœud qui est un nœud IN pour une session multicast maintient une table MFIT (*Multicast Forwarding IN Table*) indexée par le couple (S, G), où S

est l'adresse unicast de la source et G l'adresse du groupe multicast, et contient le nœud IN prédécesseur et la liste des nœuds IN successeurs.

Le principe de EM2NET est le suivant : lorsqu'un nœud X (ou un ensemble de nœuds) veut joindre une session multicast (S, G), il envoie un message JOIN directement à la source S en unicast. Lorsque S reçoit ce message, il génère un message BRANCH_ACK en réponse qui sert d'une part à acquitter le message JOIN et d'autre part à découvrir dynamiquement les nœuds IN. Sur le chemin du retour vers X (ou vers l'ensemble des nœuds désirant joindre le groupe), si le message atteint un nœud déjà membre du groupe multicast, celui-ci continue alors de le propager et aucune entrée n'est créée dans sa table MFIT. Si le message est intercepté par un nœud IN nouvellement crée (c'est-à-dire un nœud qui devient nœud de branchement ou un nouveau membre) alors une entrée pour la session (S, G) est créée et elle contient le nœud IN prédécesseur par lequel le message est arrivé. Le nœud l'informe alors par un message BRANCH_UP qu'il est devenu IN afin que celui-ci mette à jour sa liste des nœuds IN successeurs. Puis le message continue d'être propagé jusqu'à atteindre la totalité des nouveaux membres. Des messages BRANCH_UP et BRANCH_DOWN sont échangés périodiquement entre nœuds IN adjacents (vers le nœud IN prédécesseur et vers l'ensemble des nœuds IN successeurs respectivement) afin de gérer correctement la mobilité et les cassures. On voit donc que le protocole EM2NET réintroduit le principe d'état multicast aux nœuds intermédiaires en utilisant des tables MFIT. Ainsi, il assure une connectivité permanente entre les nœuds et permet de maintenir la structure d'arbre. Un temporisateur BRANCH_TIMEOUT est associé à chaque nœud IN afin de permettre les mises à jour. Les messages périodiques BRANCH_UP et BRANCH_DOWN réinitialisent ce temporisateur. Enfin, lorsqu'un membre souhaite quitter le groupe multicast, il envoie un message LEAVE à la source S en unicast. Après un temps FAILURE_WAITED_PERIOD, chaque nœud en aval de ce nœud initie une procédure JOIN vers la source afin de rejoindre le groupe. La figure 4.1 suivante illustre le chemin suivi par un paquet depuis la source S vers l'ensemble des destinations R1 à R6 ainsi que l'en-tête des différents paquets générés. On s'aperçoit

qu'au nœud S, la table MFIT contient une entrée nulle pour le nœud IN prédécesseur et N2 comme nœud IN successeur. Pour N7, le nœud IN prédécesseur est N6 (qui est un nœud de branchement) et les nœuds IN successeurs sont R3, R4 et R5 (qui sont des membres du groupe multicast) :

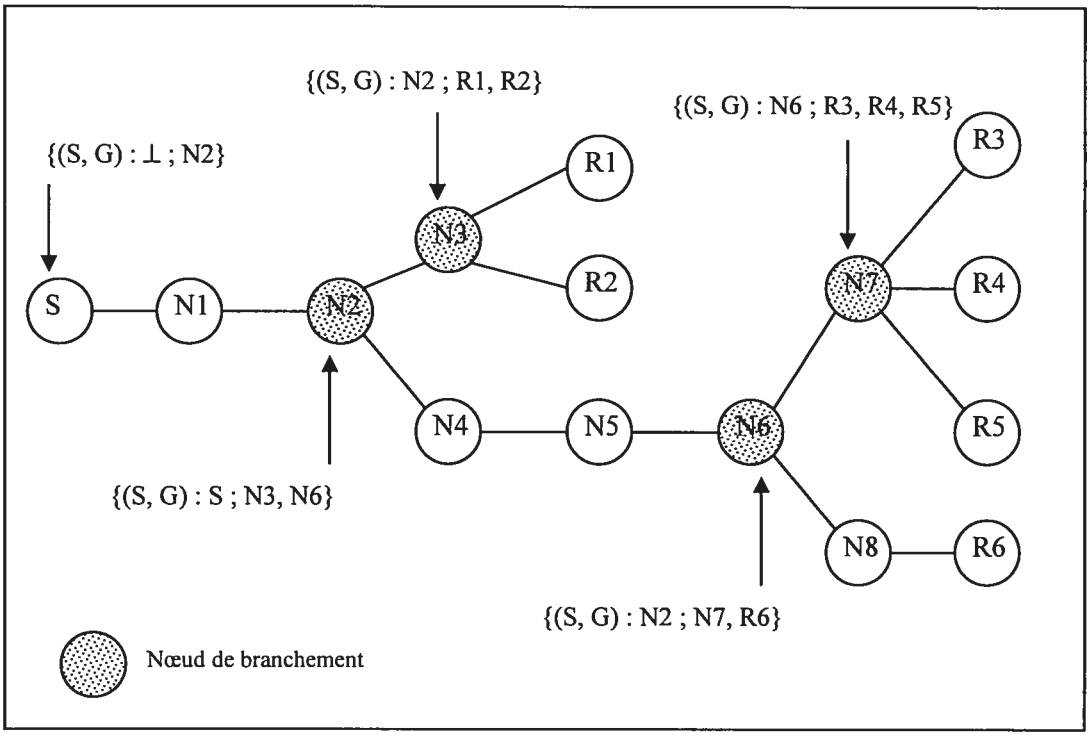

Figure 4.1. Livraison d'un paquet de données dans EM2NET.
Le contenu entre accolades représente une entrée de la table MFIT.

4.1.2 Gestion de la mobilité et des pannes

L'algorithme en pseudo code présenté sur la figure 4.2 suivante donne un aperçu de l'ensemble des opérations entreprises par un nœud X lorsqu'il se retrouve confronté au problème de la gestion de la mobilité et des pannes (non réception des messages BRANCH périodiques). Nous expliquons plus loin en détail ces opérations ainsi que la méthode de réparation de l'arbre.

```
fonction gestion_mobilité ()

entrée:
  X     // nœud initiateur de la procédure
  S     // source

début   // le temporisateur BRANCH_TIMEOUT a expiré

  si !X->recevoirBRANCH_UP(NIN(X))
  alors
    X->supprimer(NIN(X))    // table MFIT

    si !IN(X)
    alors
      X->envoi(LEAVE, S)
    fin si

  sinon si !X->recevoirBRANCH_DOWN(PIN)
  alors

    // X informe ses NIN de ne pas déclencher une procédure
    X->envoi(BRANCH_FAILURE, NIN(X))
    X->recherche_locale(TTL)

    si X->recevoirBRANCH_DOWN(Y)
    alors
      // X informe ses NIN qu'il a rejoint l'arbre
      X->envoi(BRANCH_DOWN, NIN(X))
    sinon
      X->envoi(JOIN, S)

      pour chaque nœud N := NIN(X)
      début
        N->envoi(JOIN, S)
      fin

    fin si

  fin si
fin
```

Figure 4.2. Algorithme de gestion de la mobilité et des pannes dans EM2NET.

Comme nous venons de le voir dans la section précédente, les messages BRANCH_UP et BRANCH_DOWN sont envoyés périodiquement. Ainsi, lorsqu'un nœud IN ne reçoit pas ces messages après l'expiration du temporisateur, il en déduit que son nœud IN prédécesseur ou qu'un de ses nœuds IN successeurs n'est plus joignable (déplacement, mise en veille, batterie épuisée, ...). Deux cas sont alors

possibles, comme nous pouvons le remarquer sur l'algorithme 4.1.2.1 : 1) lorsqu'il s'agit de la non réception d'un message BRANCH_UP par un nœud X, il suffit de supprimer l'entrée correspondante dans la table MFIT. Si X n'est plus un nœud IN, il quitte alors le groupe en envoyant un message LEAVE à la source ; 2) si par contre, un message BRANCH_DOWN est perdu ou n'est pas reçu par un nœud X, il faut initier une procédure de réparation afin de rejoindre l'arbre multicast. Le nœud X devient la racine d'un sous-arbre qui n'est plus accessible par la source et il doit se charger de déclencher la procédure. Celle-ci peut être décomposée en 3 étapes :

- Envoi de messages BRANCH_FAILURE à tous les nœuds en aval (afin qu'ils n'initient pas eux-mêmes une procédure de réparation) ;

- recherche locale avec un champ limité (utilisation du TTL) afin de retrouver rapidement un nœud IN dans l'arbre. Un nœud IN peut répondre à la requête par l'intermédiaire d'un message BRANCH_DOWN, celui-ci étant par la suite propagé par X à tous ses nœuds en aval. Si X reçoit plusieurs réponses, il sélectionne un nœud IN suivant certains paramètres (qui peuvent dépendre des besoins de l'application) ;

- si l'étape précédente a échoué, alors X envoie un message JOIN en unicast à la source S et l'ensemble des opérations de jointure à un groupe sont répétées.

Cette procédure peut être illustrée par le schéma 4.3 suivant. Le nœud N5 s'est déplacé et N6 n'a pas reçu les messages périodiques BRANCH_DOWN qui lui étaient adressés par N2. N6 va donc dans un premier temps informer les nœuds en aval de la cassure dans l'arbre par l'intermédiaire de messages BRANCH_FAILURE (étape 1). Puis il va tenter de rejoindre l'arbre à l'aide d'une procédure de recherche locale pour trouver un nœud IN (étape 2). L'étape 2 ayant échouée, X envoie à la source S un message JOIN afin de rejoindre le groupe (étape 3) :

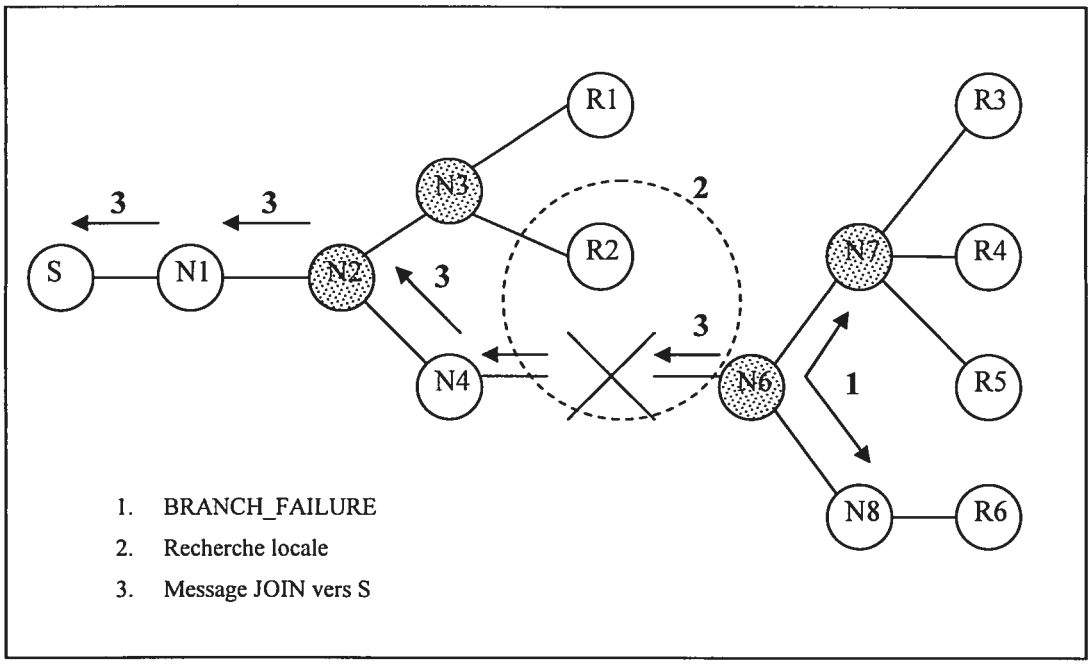

Figure 4.3. Procédure de réparation de l'arbre dans EM2NET.

4.2 Modèle de simulation et méthodologie

4.2.1 Environnement de simulation

La plateforme de simulation que nous avons utilisée est ns-2 (version 2.29) [12]. Ns-2 est un simulateur à événements discrets qui permet d'exécuter tout type de scénario sur des topologies définies par l'utilisateur. Pour visualiser les résultats, un outil graphique est disponible : NAM (*Network AniMator*). Celui-ci permet, outre la visualisation des résultats de la simulation, l'édition de scénarios simples. Ns-2 est écrit en C++ et OTcl et permet donc à l'utilisateur d'implémenter son propre protocole et de le simuler dans différents environnements. À l'origine, ns-2 a été désigné pour les réseaux filaires mais a par la suite été étendu pour supporter des simulations dans des environnements mobiles [26].

Le modèle de simulation est le suivant : un réseau MANET de 75 nœuds mobiles placés aléatoirement sur une surface de 1000m x 1000m [9, 19]. Lorsque la simulation commence, chaque nœud choisit au hasard une destination et commence à se déplacer dans sa direction à une vitesse constante. Lorsque la destination est atteinte, le nœud s'arrête pendant un certain laps de temps puis recommence le processus de déplacement. La portée de transmission radio de chaque nœud est fixée à 250 mètres et la capacité du canal est de 2 Mbits/sec. L'énergie initiale dont sont pourvus chacun des nœuds est de 10 joules. Une instance de simulation dure 600 secondes et il n'y a pas de partition du réseau au cours de la simulation. Nous utilisons dans nos expériences un modèle de propagation dit *free space* et le protocole IEEE 802.11 MAC avec DCF (*Distributed Coordination Function*) est employé comme protocole MAC [27, 28]. Il est à noter que les nœuds utilisent la réservation de canal RTS/CTS (*Request To Send/Clear To Send*) exclusivement pour les transmissions de paquets de contrôle en unicast vers des voisins spécifiques. Toutes les autres transmissions utilisent CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*). Enfin, pour les simulations qui suivent, nous avons considéré un trafic CBR (*Constant Bit Rate*) avec une taille de paquets de 512 bytes.

4.2.2 Paramètres de simulation

Nous observons les résultats des simulations en faisant varier trois paramètres. Ceci est fait afin d'identifier clairement l'impact du changement de ces paramètres sur les différents protocoles. Le cas standard est le suivant : un groupe multicast de 15 membres, une vitesse de 1 m/sec sans temps de pause et une charge de trafic de 2 pkt/sec. Pour étudier l'effet d'un certain paramètre, nous faisons varier celui-ci tandis que les autres sont fixés au cas standard. Ainsi, il est plus aisé de se concentrer sur les changements de performances résultant de la variation d'un paramètre d'intérêt. Il est important de remarquer que le cas standard a une charge de trafic et une vitesse de mobilité relativement faibles. De plus, parmi tous les nœuds

du réseau, seulement 15 d'entre eux ont été choisis pour faire partie du groupe multicast et ce, afin d'étudier les performances des protocoles dans des environnements de petite taille.

Nous faisons varier la mobilité des nœuds à l'aide de différentes vitesses de déplacement : de 0 à 10 m/sec ; le nombre de membres du groupe varie de 5 à 75 nœuds et la charge du trafic va de 1 pkt/sec à 40 pkt/sec. Les résultats que nous allons présenter sont une moyenne calculée à partir de plusieurs instances de simulation.

4.2.3 Protocoles

Les protocoles que nous avons choisi de comparer avec EM2NET sont E²M, ODMRP et MAODV. ODMRP et MAODV sont déjà disponibles dans ns-2, nous n'avons donc pas eu à les implémenter, ce qui n'est pas le cas de E²M. Étant donné que E²M est une amélioration de DDM et donc du schéma Xcast (nous avons montré cela dans le chapitre précédent), nous avons restreint nos comparaisons à E²M en ce qui concerne les protocoles multicast explicites. Nous avons également choisi de comparer EM2NET avec MAODV car celui-ci est le plus représentatif des protocoles basés sur une structure d'arbre. Enfin, ODMRP a quand à lui été sélectionné car il est, d'après de nombreuses études, le meilleur des protocoles multicast proposés à ce jour pour les réseaux ad hoc [9, 10, 25].

Lors de l'implémentation de E²M, nous avons suivi les spécifications du protocole telles que définies dans la littérature [8]. Lorsque des détails n'étaient pas indiqués clairement (valeur des différents temporisateurs par exemple), nous avons directement questionné les concepteurs du protocole à ce sujet. En ce qui concerne EM2NET, seul le principe général était décrit dans l'article qui lui était consacré [11]. Il a donc fallu définir et implémenter les structures de données, les

temporisateurs et autres spécifications nécessaires à son fonctionnement. Pour ces deux protocoles, nous avons suivi un schéma standard pour l'implémentation d'un nouveau protocole sous ns-2. Dans un premier temps, il a fallu créer une classe qui définit les en-têtes des différents paquets circulant dans le réseau (c'est ici que se fait la différence entre le codage des en-têtes dans E²M et EM2NET). Ces paquets sont utilisés pour échanger de l'information entre les objets de la simulation. À titre d'exemple, voici à quoi ressemble la définition de l'en-tête d'un paquet dans EM2NET (Cf. figure 4.4) :

```
/*****************************
 * En-tête d'un paquet EM2NET *
 *****************************/

struct hdr_em2net {
  u_int8_t  type;                  // type du paquet
  u_int8_t  pkt_forwarder;         // nœud intermédiaire
  nsaddr_t  src;                   // adresse source
  u_int8_t  previous_in;           // nœud IN prédécesseur
  u_int8_t  next_in[MAX_MEMBERS]; // liste des IN successeurs
  int       seq_no;                // numéro de séquence
  u_int8_t  num_hops;              // nombre de sauts

  ...
  ...
  ...

};
```

Figure 4.4. Définition sommaire de l'en-tête d'un paquet dans EM2NET.

Puis il a fallu implémenter les agents de routage dans une classe spécialement dédiée. Un agent dans ns-2 est maintenu par chaque nœud du réseau et est responsable de l'envoi et du traitement des paquets qu'il reçoit. Pour cela, il extraie le type du paquet reçu (message de contrôle ou message de données) ainsi que l'adresse source et détermine l'action à effectuer suivant un certain nombre de règles (dépendantes des protocoles). Le lien avec le code C++ correspondant à l'agent de

routage se fait à l'aide de scripts OTcl (définition de la topologie du réseau, des membres du groupe, de la charge du trafic et de tous les paramètres de la simulation). Enfin, une dernière classe a été créée afin de définir les structures de données et les opérations pour les différents objets de la simulation (les membres du groupe multicast, les nœuds particuliers tels que les nœuds XF dans E²M ou les nœuds IN dans EM2NET, ...). Ces opérations peuvent consister en l'ajout ou la suppression d'un membre ou bien encore la prise en charge par un nœud XF dans E²M d'un nouveau nœud.

Enfin, il nous a fallu décider quel protocole unicast serait utilisé conjointement avec E²M et EM2NET (ODMRP ne nécessite pas l'utilisation d'un protocole unicast sous-jacent pour fonctionner). Nous avions le choix entre DSR [29] et AODV [16] qui sont deux protocoles de routage « à la demande » : la découverte des routes est initiée par la source. DSR utilise un principe de cache et maintient plusieurs routes vers la destination tandis que AODV se sert de tables de routage avec une route par destination et dispose d'un mécanisme pour prévenir les boucles et déterminer l'ancienneté des routes. En se basant sur les études menées en [30, 31], nous avons opté pour AODV, qui se trouve également être le protocole de routage unicast utilisé par MAODV. En effet, il a été démontré que ce protocole obtenait de meilleures performances dans des environnements soumis à de fortes charges de trafic et à une mobilité élevée [32, 33]. Bien que AODV génère plus de surcharge de contrôle que DSR (étant donné qu'une seule route par destination est maintenue dans les tables de routage), le système de cache agressif utilisé dans ce dernier ne permet pas de disposer de routes récentes : DSR attend que la totalité des routes présentes dans le cache aient été utilisées et de ce fait, certaines d'entre elles peuvent devenir inutilisables et ainsi « polluer » le cache. Au contraire, AODV utilise une approche plus conservatrice et permet, en se basant sur le numéro de séquence, de déterminer la route la plus récente (celles n'ayant pas été utilisées récemment sont supprimées). Ns-2 fournit de base l'agent de routage AODV. Il nous a simplement fallu le modifier afin de supporter le schéma Xcast. Nous avons pour cela adopté une approche inter couches permettant à un agent Xcast d'accéder à la table de routage

de l'agent AODV. Ainsi, lorsqu'une route n'est plus disponible, l'agent Xcast se contente d'envoyer un paquet unicast à l'agent AODV, ce qui déclenche une requête de route pour la destination. Lorsqu'une réponse pour la route est reçue par l'agent AODV, celui-ci met à jour sa table de routage. L'agent Xcast peut désormais obtenir directement l'information sur le prochain nœud à partir de la table de routage de l'agent AODV. Le patron d'implémentation que nous avons adopté pour les deux protocoles peut être schématisé de la façon suivante (Cf. figure 4.5) :

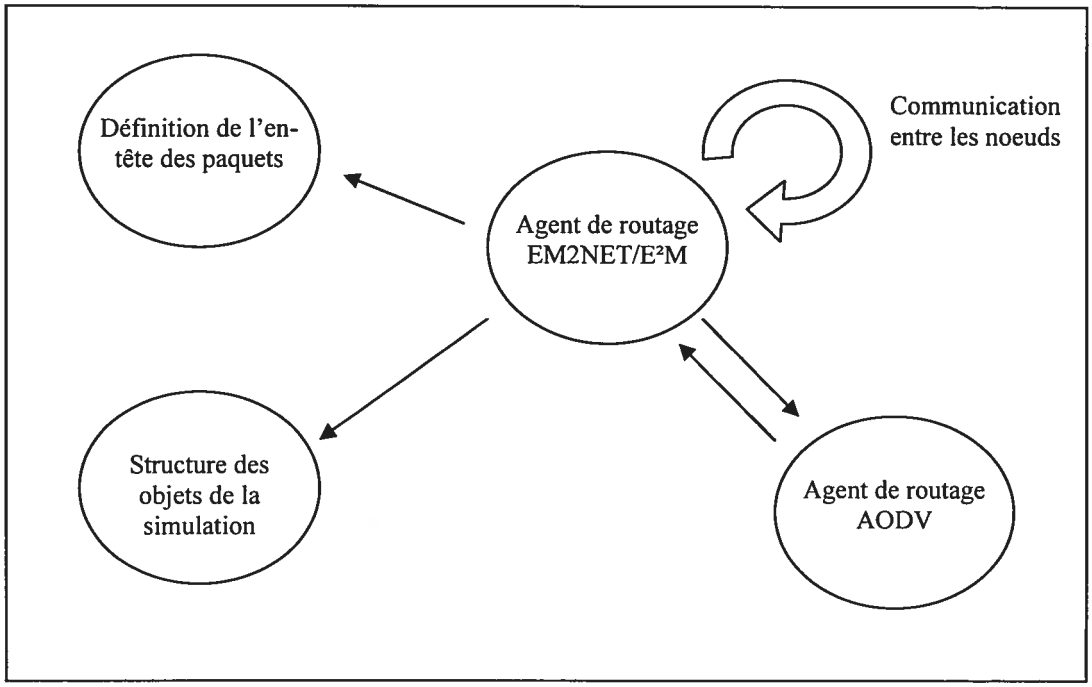

Figure 4.5. Schéma d'implémentation des protocoles E²M et EM2NET.

Finalement, le tableau 4.1 suivant synthétise les principales propriétés des protocoles que nous avons simulés. Pour chacun d'entre eux, les valeurs des paramètres reflètent leur comportement optimal, c'est-à-dire qu'ils obtiennent de meilleures performances dans ce contexte spécifique. Ces valeurs ont été calculées à des fins d'optimisation : aucun protocole n'a été favorisé par rapport à un autre [9, 19, 25].

Protocoles	Paramètres	Valeurs
ODMRP	Intervalle de rafraîchissement des messages JOIN_QUERY	3 sec
	Délai d'attente avant acquittement d'un message JOIN_QUERY	25 msec
	Nombre maximum de retransmissions des messages JOIN_REPLY	3
MAODV	GROUP_HELLO interval	5 sec
	HELLO interval	1 sec
	Route Discovery Timeout	3 sec
E²M	Seuil pour devenir XF (en nombre de nœuds)	8
	Délai d'attente pour un nœud XF après réception d'un message XF_JOIN	8 sec
	Taux de livraison des messages MEMBER_REFRESH	3 sec
EM2NET	Taux de livraison des messages BRANCH_UP et BRANCH_DOWN	5 sec
	Valeur du temporisateur BRANCH_TIMEOUT	10 sec
	Délai d'attente avant une procédure de jointure (FAILURE_WAITED_PERIOD)	5 sec

Tableau 4.1. Valeur des paramètres pour les protocoles ODMRP, E²M et EM2NET.

4.2.4 Métriques de comparaison

Nous avons utilisé un certain nombre de métriques afin de comparer les performances des différents protocoles multicast. Certaines d'entre elles sont des métriques suggérées par le groupe de travail IETF MANET [34] pour l'évaluation de protocoles de routage unicast/multicast tandis que les autres sont les métriques que nous avons considérées être les plus représentatives des performances des protocoles multicast explicites (taille moyenne de l'en-tête des paquets et énergie normalisée).

- Ratio de livraison des paquets : c'est le ratio entre le nombre de paquets de données réellement délivrés aux destinations et le nombre de paquets de données censés être reçus (si l'adhésion au groupe multicast est statique, ce

dernier est égal au nombre de paquets de données transmis par la source multiplié par le nombre de membres). Cette métrique sert à mesurer l'efficacité d'un protocole [35] ;

- Taille moyenne de l'en-tête des paquets : ce chiffre représente la taille moyenne nécessaire pour inclure la liste totale des destinations dans l'en-tête. Il est évident que plus la taille de l'en-tête est grande et plus le temps nécessaire pour traiter les paquets aux nœuds intermédiaires va être grand (découverte des adresses dans l'en-tête, création des copies multicast et transmission de gros paquets). Cela va donc influer directement sur l'énergie consommée par les nœuds. Il est donc important que la taille de l'en-tête soit la plus petite possible. Cette métrique ne s'applique qu'aux protocoles E²M et EM2NET ;

- Surcharge des bytes de contrôle : il s'agit du ratio entre le nombre de bytes de contrôle transmis et le nombre de bytes de données envoyés. Cela inclut également les bytes des en-têtes du protocole AODV (qui est utilisé dans E²M, EM2NET et MAODV). Il est à noter que les paquets de contrôle ne transportent pas de charge utile de données. Cette valeur a pour but de mesurer l'efficacité des paquets de contrôle dans la livraison des données ;

- Énergie normalisée : l'énergie étant une ressource rare dans les réseaux ad hoc mobiles, sa conservation et son utilisation efficace est un défi majeur. Il est donc primordial que les nœuds préservent le plus possible leur énergie afin d'augmenter la durée de vie du réseau. Nous comparons équitablement la consommation d'énergie des nœuds dans les trois protocoles en calculant l'énergie moyenne par paquet de données (le ratio entre la consommation totale d'énergie sur l'ensemble des nœuds du réseau et le nombre total de paquets uniques délivrés aux destinations).

4.3 Analyse théorique

4.3.1 Messages de contrôle périodiques

Dans cette section, nous allons étudier l'influence de la taille du groupe multicast sur le nombre de paquets de contrôle échangés au cours de l'exécution des protocoles EM2NET et E²M. Commençons par un bref rappel. Dans EM2NET, les messages de contrôle sont les messages BRANCH_UP et BRANCH_DOWN envoyés par un nœud IN respectivement vers son nœud IN prédécesseur et vers ses nœuds IN successeurs. Le taux de livraison de ces messages est de 1 message toutes les 5 secondes. Soit t_1 cette valeur. En ce qui concerne E²M, les nœuds XF envoient périodiquement des messages XF_REFRESH en direction de la source pour l'informer de changements potentiels dans la liste des nœuds qu'ils desservent. Les nœuds qui ne sont pas pris en charge par un nœud XF transmettent quand à eux des messages MEMBER_REFRESH directement vers la source. Les messages MEMBER_REFRESH et XF_REFRESH sont envoyés toutes les 3 secondes. Soit t_2 cette valeur. Enfin, une instance de simulation dure $t_3 = 600$ secondes. Il est à noter que les valeurs t_1, t_2 et t_3 sont conformes aux valeurs que nous avons utilisées lors de l'implémentation des protocoles.

Nous avons mentionné précédemment qu'un nœud IN est un membre du groupe multicast ou un nœud de branchement. Soit x_1 la taille du groupe multicast, on en déduit que le nombre de nœuds IN est supérieur ou égal à x_1. Afin de simplifier les calculs, plaçons-nous dans le meilleur cas et considérons x_1 comme le nombre de nœuds IN. On sait qu'un tel nœud ne peut avoir qu'un et un seul nœud IN prédécesseur (1 message BRANCH_UP). Le nombre de ses nœuds IN successeurs est quand à lui compris entre 1 et d. Là encore, nous prenons en compte le meilleur

cas qui correspond à une valeur de d égale à 1 (1 message BRANCH_DOWN). Calculons alors le nombre n de messages de contrôle BRANCH_UP et BRANCH_DOWN transmis par un nœud IN :

$$n = [x_1 \times (1+d)] \times (t_3 / t_1)$$

Lorsque x_1 est égal à 5, le nombre de paquets de contrôle échangés au cours de la simulation est égal à 1200. Si l'on considère un groupe multicast de 75 membres, cette valeur passe alors à 18 000. Nous venons de borner la valeur de n pour EM2NET.

Étudions maintenant le cas du protocole E²M. Le problème réside dans l'estimation du nombre de nœuds XF sélectionnés suivant le nombre de membres dans le groupe. Cette valeur est en effet susceptible de changer à tout instant car les nœuds sont mobiles et cela a un impact sur leur statut. Nous allons donc baser notre analyse sur les simulations que nous avons menées par la suite (section 4.4). Nous avons calculé le nombre moyen de nœuds XF en fonction de la taille du groupe multicast (Cf. figure 4.9). Les résultats sont les suivants (Cf. tableau 4.2) :

Taille du groupe multicast	Nombre de nœuds XF
5	0
10	0
20	2
30	3
40	3
50	4
60	5
75	6

Tableau 4.2. Nombre de nœuds XF en fonction de la taille du groupe multicast.

De plus, nous allons considérer le pire cas dans l'attribution du statut de nœud XF, c'est-à-dire qu'un nœud XF dessert exactement 8 membres. En effet, s'il est responsable de plus de 8 nœuds, il y aura encore moins de paquets de contrôle transmis. Par exemple, pour un groupe multicast de 60 membres, il y a 5 nœuds XF qui s'occupent chacun de 8 membres et il reste donc 20 membres isolés (non pris en charge par un nœud XF). Il est désormais possible d'estimer le nombre n de messages de contrôle échangés dans E²M. Soit x_1 la taille du groupe multicast et x_2 le nombre de nœuds XF correspondant :

$$n = [x_1 - (x_2 \times 8) + x_2] \times (t_3 / t_2)$$

Pour une valeur de x_1 égale à 5, cela donne un résultat de 1000 paquets de contrôle échangés. Lorsque x_1 est égal à 75, ce sont maintenant 6600 messages qui sont transmis périodiquement. La valeur de n en fonction de la taille du groupe multicast est donc comprise entre 1000 et 6600.

Avant de tirer les conclusions de cette étude, il est important de souligner que ces résultats ne prennent pas en compte les possibles pertes de paquets de contrôle qui peuvent être engendrées par les collisions, les dépassements de tampons ou bien encore par la mobilité des nœuds. L'impression générale qui se dégage de l'analyse menée est que le nombre de paquets de contrôle échangés est plus important dans EM2NET que dans E²M. En effet, les nœuds qui sont XF dans E²M se chargent de transmettre vers la source les messages périodiques à la place des nœuds dont ils sont responsables et cela permet d'en diminuer considérablement le nombre. Ainsi, il est intéressant de remarquer que pour une taille de groupe multicast faible, le nombre de paquets de contrôle transmis est pratiquement équivalent dans les deux protocoles étant donné qu'il n'y a pas de nœuds XF sélectionnés dans E²M alors que la différence est très nettement visible pour un groupe multicast très dense. Cette impression est d'autant plus significative que nous avons considéré le meilleur cas pour EM2NET et le pire pour E²M. Cependant, nous verrons lors des simulations que

les messages de contrôle périodiques dans EM2NET permettent de maintenir efficacement la connectivité de l'arbre, de s'affranchir du problème de la mobilité des nœuds et ainsi d'offrir un meilleur ratio de livraison des paquets.

4.3.2 Gestion de la mobilité des noeuds

Intéressons-nous à présent à la gestion de la mobilité dans les protocoles E²M et EM2NET. Nous allons comparer et identifier le temps et les opérations nécessaires pour qu'un nœud rejoigne l'arbre multicast après s'être déplacé.

Lors de l'utilisation du protocole E²M, la source est informée du déplacement d'un nœud XF lors de la réception d'un paquet d'erreur de route généré par le protocole de routage unicast. Dans le cas du protocole AODV, si la requête de route RREQ échoue un certain nombre de fois (RREQ_RETRIES), un message d'erreur est délivré à l'application. Généralement, le nombre de retransmissions autorisées est fixé à 3 et celles-ci sont espacées de 10 secondes (ce sont les valeurs standards du protocole AODV et celles que nous avons utilisées dans nos simulations). Dès lors, la source obtient la liste des destinations servies par le nœud XF et les encode explicitement dans l'en-tête des paquets de données. Les nœuds rejoignent donc l'arbre lors de la transmission du prochain paquet. Durant ce laps de temps d'environ 30 secondes (le temps de transmission d'un paquet est de l'ordre de quelques millisecondes), de nombreux paquets peuvent avoir été transmis, dépendamment de la charge du trafic. Quand au nœud XF en question, il ne pourra rejoindre l'arbre que suite à la transmission et à l'acquittement d'un paquet MEMBER_JOIN dirigé vers la source.

Outre le délai d'attente et par conséquent la baisse du ratio de livraison des paquets engendrée par le mouvement d'un nœud XF, il convient d'en analyser les conséquences du point de vue des autres métriques. Soit k le nombre de receveurs

servis par le nœud XF, l'en-tête du paquet augmente de (k * 4 bytes) lorsque celui-ci se déplace (la source encode les k receveurs dans l'en-tête du nouveau paquet multiplié par la taille d'une adresse, généralement 4 bytes). De plus, le nœud anciennement XF ne l'est plus. Tout le bénéfice engendré par la formation de nœuds XF dans le protocole E²M est donc annihilé lorsque la mobilité est introduite.

Lorsqu'il s'agit du déplacement d'un nœud quelconque, la non réception d'un message MEMBER_REFRESH par le nœud XF auquel il est rattaché engendre la transmission d'un message XF_REFRESH dirigé vers la source et indiquant les changements dans les nœuds desservis. Dans les simulations que nous avons menées (et d'après les valeurs indiquées par les concepteurs du protocole), un message MEMBER_REFRESH est transmis toutes les 3 secondes. Cette valeur est exagérément grande afin d'éviter la transmission d'un trop grand nombre de messages de contrôle mais en contrepartie, cela augmente fortement le nombre de paquets non reçus. En effet, si l'on se place dans le pire cas, à savoir qu'un nœud se déplace immédiatement après avoir transmis un message MEMBER_REFRESH, il faudra attendre 3 secondes avant que le nœud XF auquel il est rattaché s'en rende compte.

Plaçons-nous maintenant dans le cas du protocole EM2NET. Les messages BRANCH_UP et BRANCH_DOWN sont échangés périodiquement entre les nœuds IN toutes les 5 secondes. Par conséquent, si un des nœuds IN successeurs n'est plus joignable, c'est-à-dire qu'un nœud X n'a pas reçu de messages BRANCH_UP de sa part, il en est informé immédiatement au terme des 5 secondes et il peut effectuer les opérations nécessaires (mise à jour de la table MFIT, vérification du statut de nœud IN, ...). Dans ce cas, il n'y a pas de modification dans la topologie de l'arbre.

Si l'on se place maintenant dans la situation où un nœud X ne reçoit pas de messages BRANCH_DOWN (ce qui signifie que le nœud IN prédécesseur n'est plus dans son champ d'écoute), la procédure de réparation de l'arbre est alors déclenchée et X est la racine d'un sous-arbre séparé de l'arbre multicast initial. La première

étape consiste à envoyer des messages BRANCH_FAILURE aux nœuds IN successeurs. La seconde opération a pour but de trouver un nœud IN dans l'arbre en utilisant une recherche locale avec un TTL limité. Dans nos simulations, nous avons adapté le champ TTL en fonction de la taille du groupe multicast (plus le nombre de membres dans la session multicast est grand et plus le TTL est large). Les tests que nous avons effectués ont montré qu'une réponse parvenait au nœud initiateur de la procédure en moyenne au bout de 250 millisecondes. Dans le pire des cas (échec de la recherche locale), le nœud rejoint l'arbre en envoyant un message JOIN dirigé vers la source. Il est intéressant de constater que la plupart des opérations relatives à la procédure de réparation de l'arbre sont des transmissions de messages qui sont des opérations très rapides (de l'ordre de quelques millisecondes comme nous l'avons mentionné). De plus, cette procédure aboutit à une configuration d'arbre identique à celle qu'elle était avant le déplacement des nœuds. En effet, un nœud reste un nœud IN s'il l'était auparavant : il possède désormais un nouveau nœud IN prédécesseur tandis que ses nœuds IN successeurs sont informés de la nouvelle situation par l'intermédiaire de messages BRANCH_DOWN. Par conséquent, le mouvement des nœuds n'a aucune incidence sur la taille moyenne de l'en-tête des paquets de données.

Nous venons de montrer que la gestion de la mobilité des nœuds dans EM2NET est particulièrement efficace dans le sens où elle aboutit rapidement à une reconstruction de l'arbre sans affecter les performances du protocole. Cette réactivité vis-à-vis de la mobilité se fait au détriment de la surcharge des paquets de contrôle. En effet, entre les messages BRANCH_FAILURE, la recherche locale et la réactivation des routes par des messages BRANCH_DOWN, le nombre de paquets de contrôle échangés est élevé lors de la procédure de réparation de l'arbre. Cependant, cela permet de maintenir un ratio de livraison de paquets élevé dans la mesure où le réseau n'est que très peu longtemps partitionné, ce qui n'est pas le cas dans le protocole E^2M.

4.3.3 Taille moyenne de l'en-tête

Finalement, nous allons étudier l'influence des caractéristiques des noeuds sur la taille moyenne de l'en-tête des paquets de données. Nous avons mentionné précédemment que cette métrique avait une importance toute particulière dans les protocoles multicast explicites car elle agit directement sur la consommation d'énergie des nœuds. Pour ODMRP et MAODV, cette valeur est toujours nulle.

Plaçons-nous dans le cas d'un arbre binaire. Pour E²M, soit d la distance d'un nœud XF à la source et h la hauteur de l'arbre en nombre de sauts. Le nombre n de destinations qui peuvent être encodées dans l'en-tête au nœud XF est égal à :

$$n = \sum_{d \le i \le h-1} 2^{i}$$

Or, pour EM2NET, cette valeur est toujours égale à 2 pour chacun des nœuds IN dans l'arbre (excepté les feuilles) car un nœud n'encode que la liste des nœuds IN successeurs dans l'en-tête. Étudions maintenant le cas général. Soit d la distance d'un nœud X vers la source. Supposons que ce nœud est un nœud XF dans E²M et qu'il sert k receveurs (le nombre requis pour devenir XF). La taille de l'en-tête d'un paquet au nœud X est k pour E²M (multiplié par la taille d'une adresse, généralement 4 bytes) tandis que pour EM2NET, plus les receveurs sont loin du nœud X et plus la taille de l'en-tête du paquet diminue (en accord avec le nombre de noeuds IN successeurs de X).

Cela montre l'équilibre entre les nœuds IN dans EM2NET et la capacité du protocole à mieux supporter les opérations de multicast et la conservation de l'énergie. Dans E²M, les nœuds XF vont être dépourvus d'énergie bien avant les autres nœuds et cela pourrait causer une fragmentation du réseau.

4.4 Résultats des simulations

4.4.1 Taille du groupe multicast

A. Scénarios

Nous faisons varier le nombre de membres dans le groupe multicast. Tandis que la vitesse de mobilité est fixée à 1 m/sec sans temps de pause et que le taux de livraison des données est de 2 pkt/sec, la taille du groupe multicast varie de 5 à 75 membres (soit la totalité des nœuds dans le réseau).

B. Résultats et analyses

L'efficacité des différents protocoles pour plusieurs tailles de groupe multicast est exposée à la figure 4.6. ODMRP n'est pas affecté par le nombre de membres dans la session multicast. Étant donné que le maillage devient dense avec l'augmentation du nombre de membres, de plus en plus de routes redondantes apparaissent et cela conduit à de meilleures performances. Au contraire, le ratio de livraison des paquets de données pour E²M et EM2NET diminue lorsque la taille du groupe augmente. Cependant, il est intéressant d'observer que pour un groupe multicast de faible densité (de 5 à 15 membres), EM2NET et ODMRP ont des ratios de livraison de paquets relativement semblables. De plus, EM2NET se comporte d'une bien meilleure manière que E²M et cela s'explique de deux façons. Premièrement, par sa capacité à prendre en compte la mobilité des nœuds, EM2NET délivre plus de paquets aux destinations, même en cas de forte mobilité. Alors que dans E²M, si un nœud XF bouge, la probabilité qu'un ou plusieurs nœuds se retrouvent isolés dans l'arbre augmente fortement. Deuxièmement, EM2NET permet un balancement de la charge entre les nœuds IN alors que E²M concentre la majorité

du trafic sur les nœuds XF : il est donc à prévoir que plus de paquets seront perdus par collision ou congestion du réseau. Dans le cas de MAODV, le ratio de livraison des paquets diminue légèrement au fur et à mesure que la taille du groupe multicast augmente. En effet, le nombre de messages de contrôle transmis s'en trouve ainsi augmenté et cela résulte en des pertes de paquets dues à de nombreuses collisions. Il convient de remarquer que EM2NET obtient de meilleures performances que MAODV et ce, même pour des groupes multicast denses. En effet, les deux protocoles sont basés sur une structure d'arbre mais EM2NET, parce qu'il utilise un mécanisme de réparation de l'arbre, maintient tout de même un ratio de livraison supérieur à celui de MAODV lorsque le groupe multicast gagne en densité.

Le nombre de bytes de contrôle transmis par rapport au nombre de bytes de données envoyés est présenté sur la figure 4.7. Les paquets de contrôle sont les paquets générés par le protocole pour établir et maintenir les informations d'appartenance au groupe. Nous remarquons que ODMRP est le moins performant des protocoles pour ce critère lorsque la taille du groupe multicast augmente. En effet, les informations d'appartenance au groupe ainsi que les routes multicast sont établis et maintenus par la source sur demande, via un mécanisme d'inondation du réseau. Les paquets de contrôle sont ainsi propagés par chacun des nœuds appartenant au *Forwarding Group* au sein du maillage. En ce qui concerne MAODV, l'augmentation de la taille du groupe entraîne une augmentation du nombre de messages de contrôle échangés mais ceux-ci ne sont pas transmis le long de multiples chemins car MAODV utilise une structure d'arbre. Cela résulte en une surcharge globalement inférieure à celle de ODMRP. Dans E²M, certains des paquets de contrôle sont pris en charge par les nœuds XF (E²M empêche le problème de JOIN_IMPLOSION). Ainsi, au fur et à mesure que le nombre de membres augmente, le nombre de nœuds XF augmente (Cf. figure 4.9) et le nombre de paquets de contrôle injectés dans le réseau se trouve ainsi diminué par rapport à EM2NET. Ces résultats sont en conformité avec ceux que nous avons obtenus lors de l'analyse théorique. Cependant, il faut noter que les messages périodiques de EM2NET sont nécessaires car ils aident à maintenir un meilleur ratio de livraison des données.

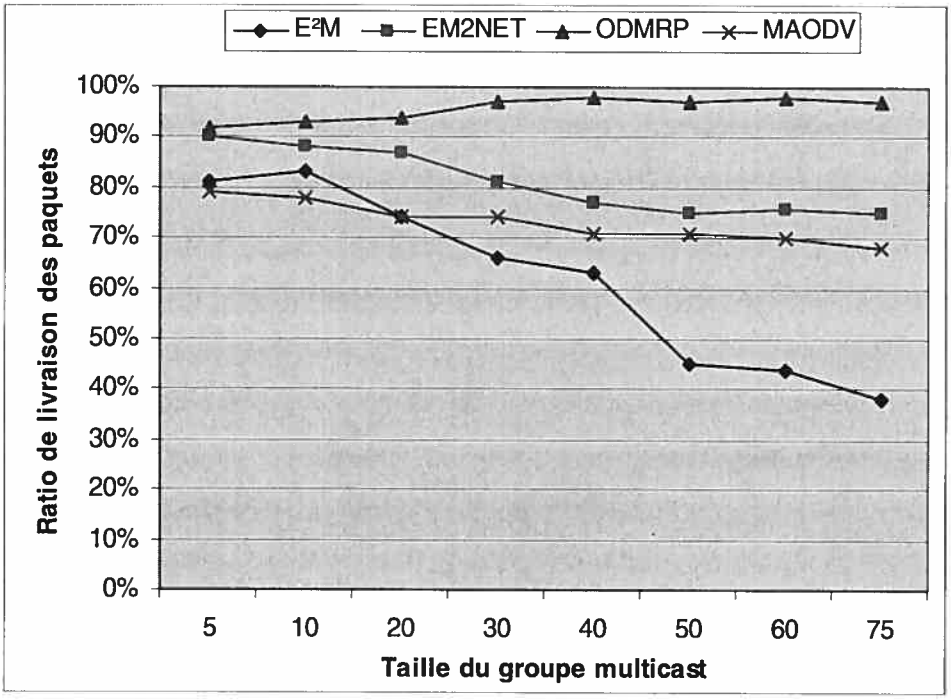

Figure 4.6. Ratio de livraison des paquets de données en fonction de la taille du groupe multicast.

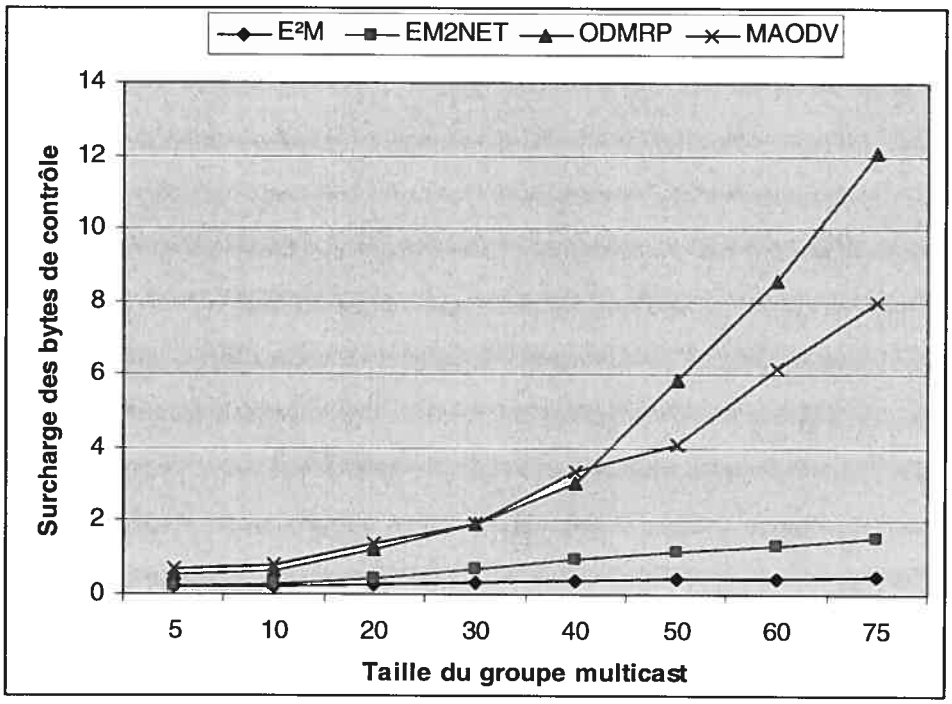

Figure 4.7. Surcharge des bytes de contrôle en fonction de la taille du groupe multicast.

Un des principaux points auxquels nous souhaitions nous intéresser lors des simulations était la taille moyenne de l'en-tête des paquets transmis dans le réseau. La signification de ce résultat est la suivante : lorsque cette valeur augmente, il y a un plus grand nombre de destinations mentionnées dans l'en-tête et cela augmente à la fois le temps pour traiter le paquet aux nœuds intermédiaires mais également la consommation d'énergie. La taille moyenne de l'en-tête des paquets pour E²M et EM2NET en fonction de la taille du groupe multicast est montrée sur la figure 4.8. Pour ODMRP et MAODV, cette valeur est toujours 0. Nous remarquons que pour un groupe peu dense, E²M se comporte de manière similaire à EM2NET étant donné qu'il n'y a pas de nœuds XF sélectionnés. Mais au fur et à mesure que le nombre de membres dans le groupe augmente, la taille de l'en-tête augmente également que ce soit pour E²M ou EM2NET. Cependant, bien que la sélection de nœuds XF aide à réduire la liste des destinations dans E²M (pour 75 membres, la taille de l'en-tête serait 75 x 4 bytes = 300 bytes pour le schéma Xcast), EM2NET se comporte d'une meilleure façon lorsque la taille du groupe multicast augmente. Cela peut être expliqué par la valeur de seuil choisie dans E²M pour la sélection d'un nœud XF.

Nous avons calculé le nombre de nœuds XF sélectionnés en fonction de la taille du groupe. La figure 4.9 illustre les résultats de cette étude. On observe que pour un seuil de sélection de 8 membres, le nombre de nœuds XF est de 0 pour 10 membres et il n'est que de 6 pour 75 membres. Une valeur de seuil plus importante entraînerait une taille d'en-tête plus grande car il serait plus difficile pour un nœud de devenir XF alors qu'une valeur plus petite causerait une importante consommation d'énergie (plus de nœuds deviendraient XF). Avec la méthode proposée par EM2NET, ce problème est résolu grâce à un balancement de la charge entre les nœuds IN : un nœud qui est un membre du groupe ou un nœud de branchement devient un nœud IN. Ainsi, plus les receveurs sont loin d'un nœud IN et plus la taille de l'en-tête à cet IN diminue (dépendamment du nombre de nœuds IN directement successeurs). La source encode seulement le nœud IN prédécesseur et les nœuds IN successeurs dans l'en-tête du paquet. Cela montre l'équilibre entre les nœuds pour supporter les opérations de multicast et ainsi mieux conserver leurs batteries.

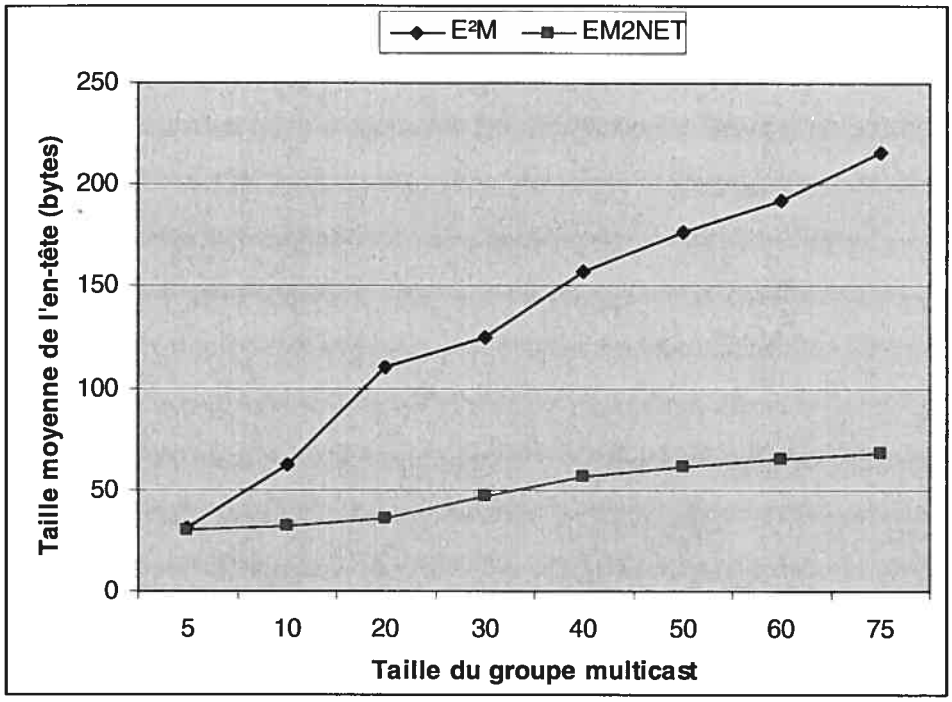

Figure 4.8. Taille moyenne de l'en-tête des paquets en fonction de la taille du groupe multicast.

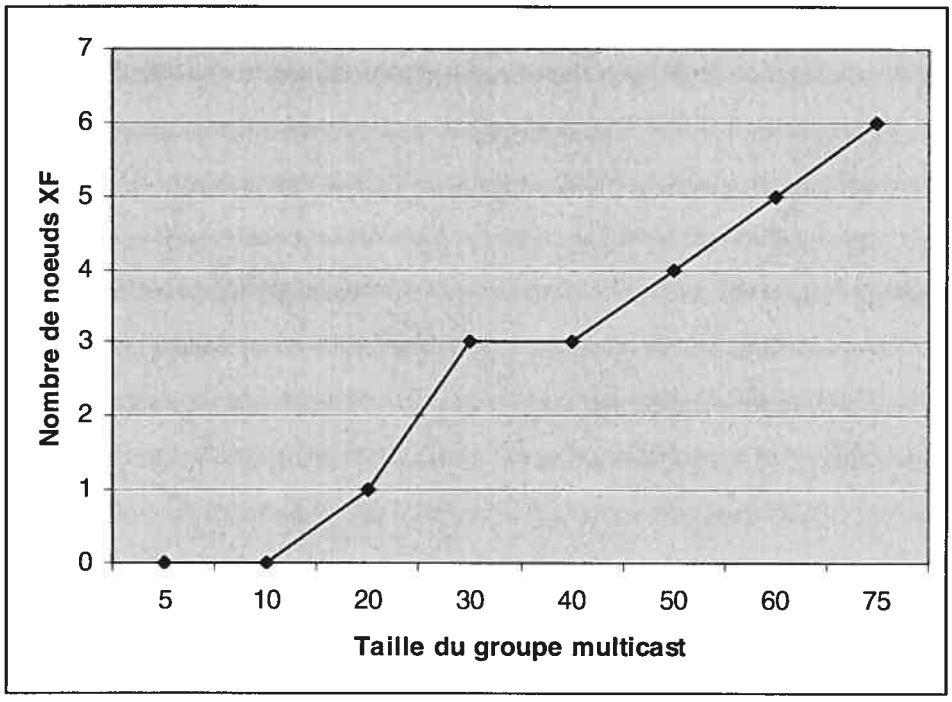

Figure 4.9. Nombre de nœuds XF dans E²M en fonction de la taille du groupe multicast.

Nous pouvons voir sur la figure 4.10 que les nœuds qui sont XF dans E²M consomment plus d'énergie car ils sont responsables de l'encapsulation, de la duplication et de la transmission des copies de paquets multicast aux nœuds qu'ils desservent (la courbe pour les nœuds XF ne débute que pour une abscisse de 20 étant donné qu'il n'y a pas de nœuds XF sélectionnés pour une taille de groupe multicast inférieure à 20 ; Cf. figure 4.9). De plus, nous avons montré lors de l'analyse théorique que la taille moyenne de l'en-tête des paquets était nettement supérieure dans E²M (les résultats ont été confirmés par les simulations précédentes), ce qui engendre une importante dépense d'énergie. Cela est particulièrement critique étant donné que la durée de vie du réseau dépend fortement des niveaux d'énergie de chaque nœud. Il se pourrait donc qu'il y ait une fragmentation du réseau dans E²M. Comme montré précédemment, EM2NET permet un balancement de la charge entre les nœuds IN et est donc moins sujet à ce phénomène.

La figure 4.11 rend compte de la perte d'énergie (exprimée en joules) en fonction de la taille du groupe multicast. Comme une conséquence de la taille moyenne de l'en-tête des paquets, on observe que les nœuds dans E²M consomment plus d'énergie que ceux dans EM2NET. Étant donné que le modèle d'énergie dans ns-2 est basé sur la puissance et le temps de transmission/réception des paquets, il est évident que les nœuds dans E²M perdront leur énergie plus rapidement que dans EM2NET. Pour ODMRP, le grand nombre de paquets de contrôle échangés périodiquement provoque une forte consommation d'énergie : un nœud mobile verra sa batterie se vider rapidement à cause de l'inondation dans le réseau d'un trop grand nombre de messages. MAODV se comporte quand à lui d'une meilleure façon que ODMRP mais cela s'explique par un plus faible ratio de livraison des paquets. Nous observons également que EM2NET consomme moins d'énergie que MAODV tout en délivrant plus de paquets à leurs destinations.

Afin de comparer équitablement les quatre protocoles, étudions maintenant l'énergie normalisée en fonction de la taille du groupe multicast. L'énergie normalisée représente la consommation d'énergie en tenant compte du nombre de

paquets délivrés. Les résultats sont présentés sur la figure 4.12. Pour un groupe multicast de faible densité, la différence entre EM2NET et ODMRP est considérable. En effet, ils possèdent des ratios de livraison de paquets relativement équivalents mais EM2NET consomme beaucoup moins d'énergie lors du processus de livraison des données, d'où l'écart important entre les deux protocoles. Au fur et à mesure que la taille du groupe multicast augmente, cet écart diminue car ODMRP consomme toujours beaucoup d'énergie mais maintient un meilleur ratio de livraison des paquets. Cependant, l'énergie normalisée de EM2NET est tout le temps inférieure à celle de ODMRP, quelle que soit la taille du groupe. Si l'on considère maintenant le cas de MAODV, nous observons une légère diminution de l'énergie normalisée lorsque la taille du groupe multicast augmente. Cependant, cette valeur est toujours supérieure à EM2NET car MAODV maintient un ratio de livraison des paquets relativement faible et ce, même pour des groupes multicast larges. Nous pouvons conclure en disant que EM2NET préserve la durée de vie du réseau d'une façon plus efficace que les protocoles multicast actuels proposés pour MANETs.

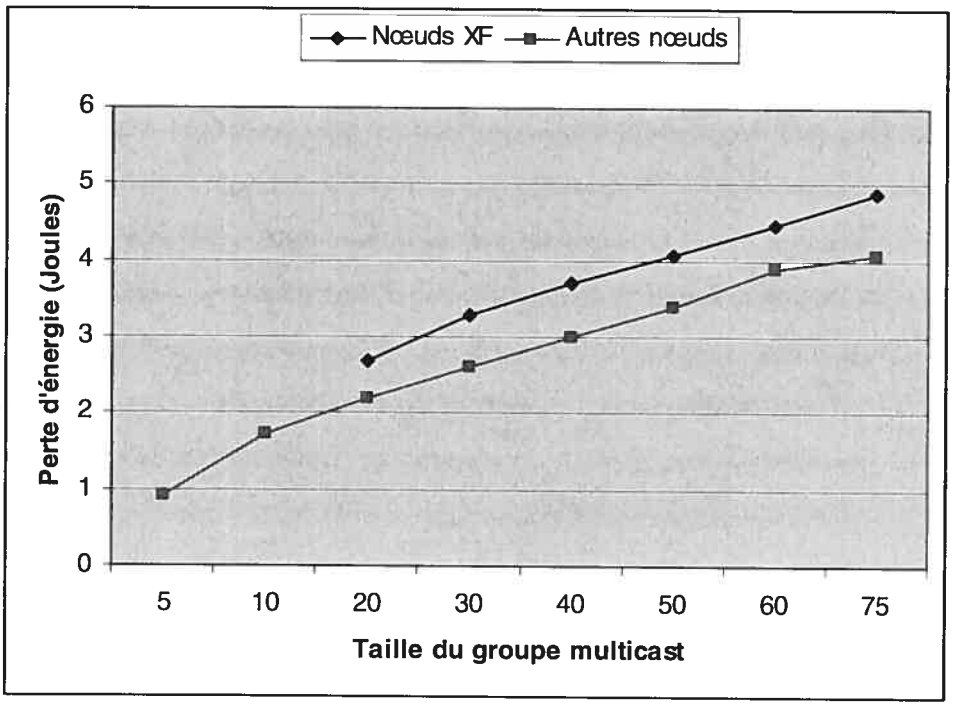

Figure 4.10. Perte d'énergie des noeuds dans E²M en fonction de la taille du groupe multicast.

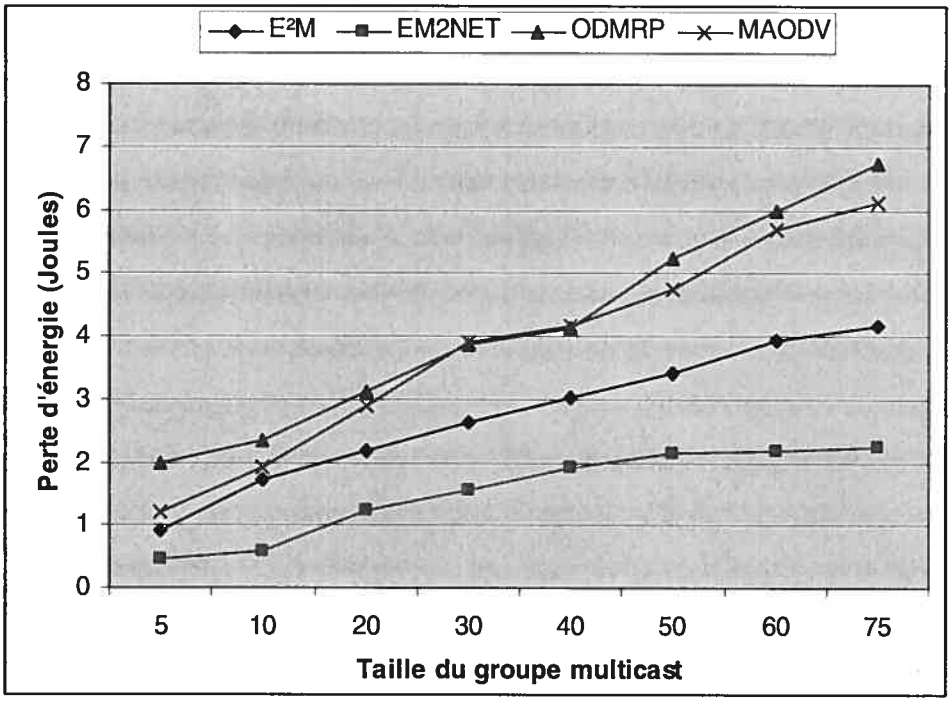

Figure 4.11. Perte d'énergie en fonction de la taille du groupe multicast.

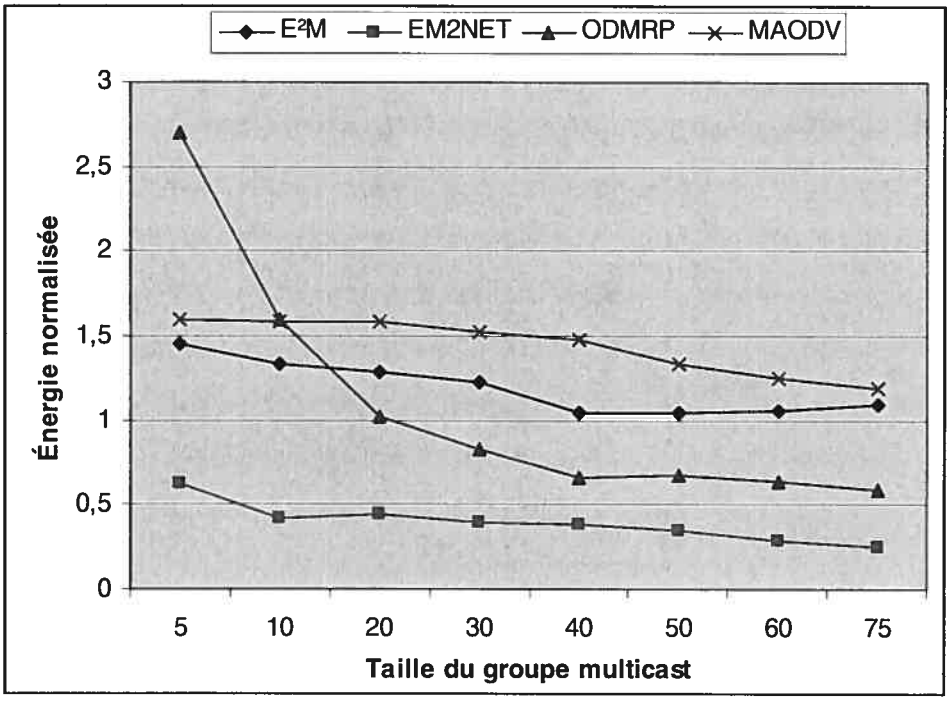

Figure 4.12. Énergie normalisée en fonction de la taille du groupe multicast.

4.4.2 Vitesse de mobilité

A. Scénarios

Pour cette expérience, nous faisons varier la vitesse de déplacement des nœuds de 0 à 10 m/sec sans temps de pause tandis que les autres paramètres sont fixés au cas standard (15 membres dans la session multicast et un taux de livraison de données de 2 pkt/sec). Les directions de déplacement sont sélectionnées aléatoirement. Lorsqu'un nœud atteint la frontière du terrain, il rebondit et continue de se déplacer à vitesse constante.

B. Résultats et analyses

La figure 4.13 est une illustration du ratio de livraison des paquets des quatre protocoles sous différents scénarios de mobilité. On s'aperçoit que ODMRP est très performant même dans des situations extrêmement dynamiques (vitesse de 10 m/sec). En effet, la topologie maillée fournit des routes redondantes et les chances pour un paquet d'atteindre sa destination ne se trouvent pas affectées par la vitesse de mobilité des nœuds. Il convient donc de dire que ODMRP est robuste à la mobilité (aussi efficace que la diffusion). Dans MAODV, l'augmentation de la vitesse de mobilité cause de fréquents changements de liens étant donné que le protocole repose sur un chemin unique dans l'arbre. Afin de prévenir les informations de route persistantes, MAODV doit reconfigurer l'arbre plus souvent, ce qui résulte en une baisse du ratio de livraison des paquets. Pour chacun des deux autres protocoles, E²M et EM2NET, le ratio de livraison diminue au fur et à mesure que la vitesse des nœuds augmente. Il y a plusieurs facteurs qui contribuent à ces performances lorsque la mobilité est introduite. Parmi ceux-là, citons les collisions de paquets au niveau de la couche MAC étant donné que les paquets sont transmis sans RTS/CTS. Cependant, il est intéressant d'observer que le ratio de livraison pour E²M est nettement inférieur à celui de EM2NET et ceci est dû au fait qu'il ne prend pas en compte la mobilité des

nœuds. De plus, la procédure de réparation de l'arbre dans EM2NET privilégie une recherche locale en premier et contribue par conséquent à une réparation de l'arbre plus rapide (comme nous l'avons montré durant l'analyse théorique). On peut donc en conclure que EM2NET, en prenant correctement en compte la mobilité des nœuds, surpasse les protocoles multicast explicites proposés à ce jour pour les réseaux ad hoc et aide à atténuer la baisse du ratio de livraison des paquets engendrée par une forte mobilité.

Bien sûr, EM2NET a une surcharge des bytes de contrôle supérieure à celle de E²M lorsque la mobilité est élevée. En effet, aucune mise à jour n'est engendrée par le mouvement d'un nœud dans E²M. Toutefois, la procédure de réparation de l'arbre privilégie une recherche locale et ainsi, un nœud ne reste pas isolé très longtemps dans EM2NET (Cf. analyse théorique, section 4.3). De plus, ces messages de contrôle sont nécessaires car ils aident à maintenir la connectivité du groupe et contribuent ainsi à offrir un ratio de livraison des paquets plus élevé. Quand à ODMRP, la surcharge engendrée par la maintenance de la structure de maillage est très importante et est nettement supérieure à celle de EM2NET et E²M.

La figure 4.14 montre l'impact de la mobilité des nœuds sur la taille moyenne de l'en-tête des paquets. Comme nous l'avons dit plus haut, lorsqu'un nœud XF se déplace dans E²M, la source encode explicitement la liste de toutes les destinations servies par cet XF dans l'en-tête. Cela en augmente considérablement la taille, conformément à ce que nous avons montré au cours de l'analyse théorique. De plus, il sera difficile pour un nœud d'obtenir le statut de nœud XF étant donné qu'il n'y a que 15 membres dans la session multicast. La réduction significative de la taille de l'en-tête des paquets engendrée par la sélection de nœuds XF n'aura donc probablement pas lieu. Pour cette métrique, EM2NET n'est pas affecté par la mobilité étant donné que les nœuds cherchent à rejoindre l'arbre le plus rapidement possible en essayant de trouver un nœud IN localement. On peut même remarquer que la taille moyenne de l'en-tête demeure relativement constante dans EM2NET et ce, même lorsque la mobilité est élevée.

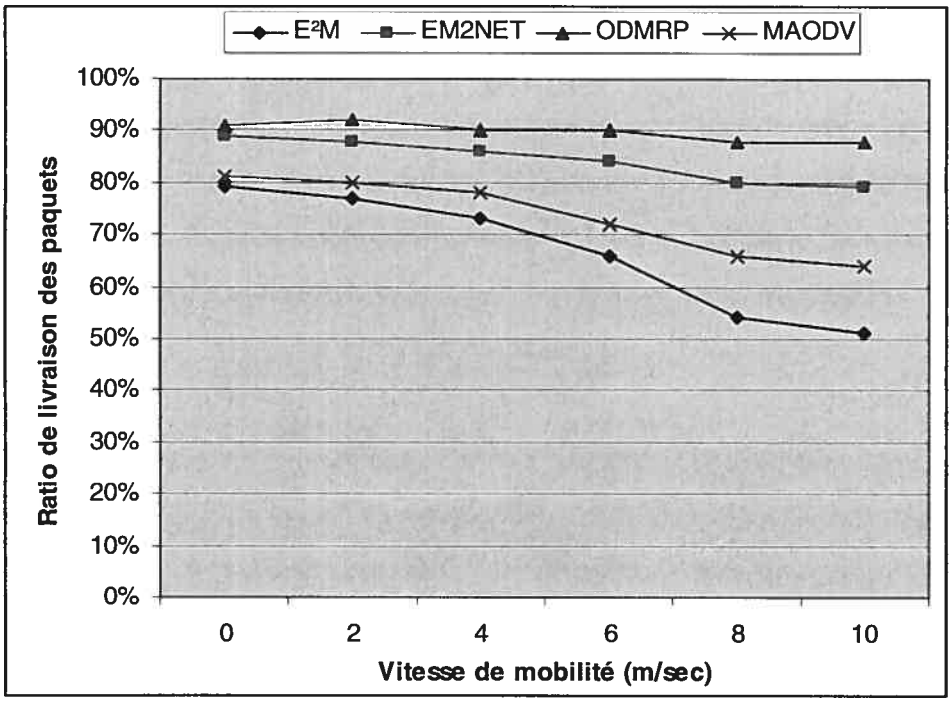

Figure 4.13. Ratio de livraison des paquets de données en fonction de la vitesse de mobilité.

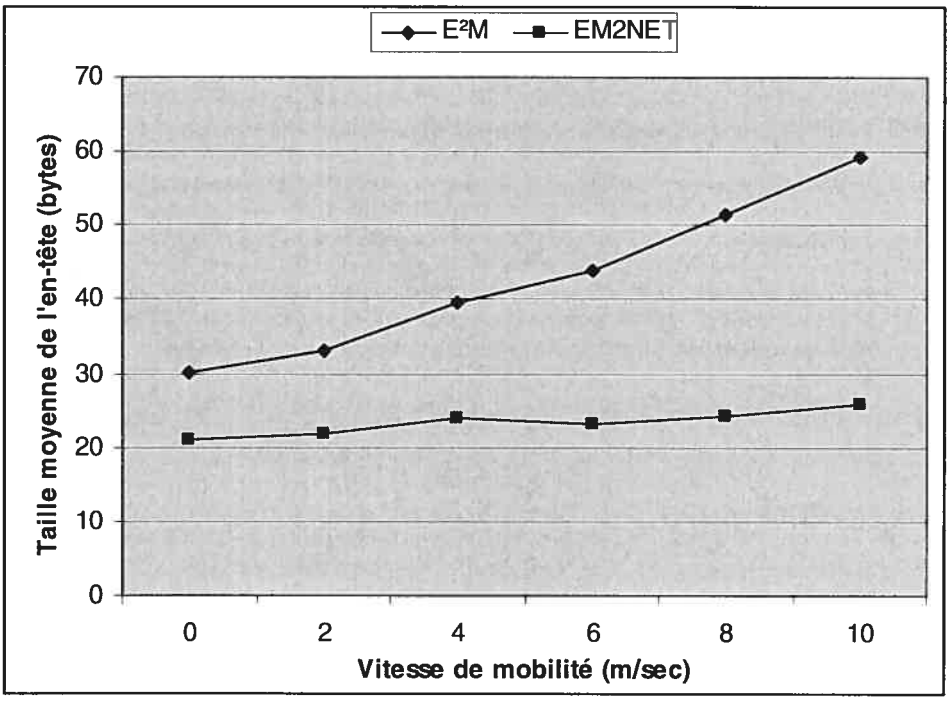

Figure 4.14. Taille moyenne de l'en-tête des paquets en fonction de la vitesse de mobilité.

Attardons-nous maintenant sur l'énergie moyenne consommée par paquet de données en fonction de la vitesse de déplacement des nœuds. La figure 4.15 nous montre comment le ratio de livraison des paquets des quatre protocoles influe sur l'énergie normalisée. Premièrement, nous observons que EM2NET obtient de meilleures performances que E²M quel que soit le scénario de mobilité utilisé. En effet, comme expliqué précédemment, la taille moyenne de l'en-tête des paquets de données augmente considérablement lorsque la vitesse de mobilité augmente, ce qui entraîne une forte consommation d'énergie. De plus, E²M a un ratio de livraison de paquets très faible lorsque la mobilité est introduite. Au contraire, EM2NET maintient une taille moyenne d'en-tête des paquets relativement constante et ne subit pas autant que E²M la mobilité des nœuds.

Un autre point intéressant sur lequel il convient de s'attarder concerne l'énergie normalisée de ODMRP. Nous pouvons voir que la courbe concernée (Cf. figure 4.15) a toujours une allure descendante au fur et à mesure que la mobilité augmente mais la réduction n'est pas aussi significative que lorsque nous avons varié la taille du groupe multicast. En effet, EM2NET offre un ratio de livraison de paquets correct même lorsque la vitesse est à son maximum car il emploie un mécanisme de réparation de l'arbre. Ainsi, l'énergie moyenne par paquet de données demeure constante. ODMRP a également un bon ratio de livraison de paquets mais les nœuds du réseau consomment plus d'énergie, d'où l'écart de performances entre EM2NET et ODMRP.

Pour MAODV, la baisse sévère du ratio de livraison des paquets au fur et à mesure de l'augmentation de la vitesse de déplacement des nœuds entraîne une énergie normalisée conséquente. En effet, il consomme plus d'énergie que EM2NET et E²M tout en permettant une augmentation du nombre de messages de contrôle échangés.

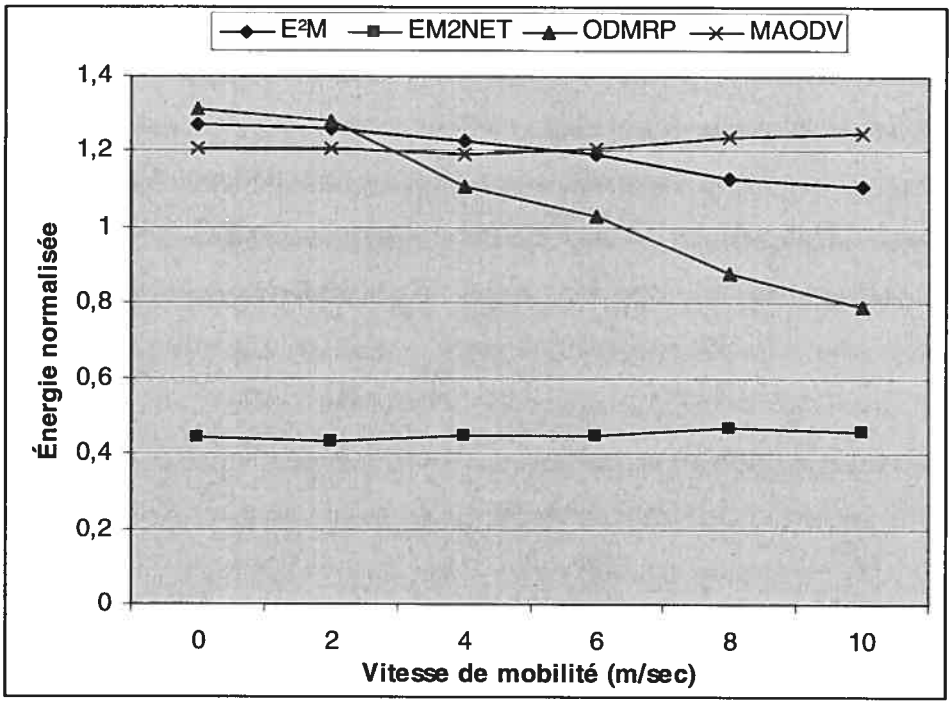

Figure 4.15. Énergie normalisée en fonction de la vitesse de mobilité.

4.4.3 Charge du trafic

A. Scénarios

Afin d'étudier l'impact de la charge du trafic sur les performances des protocoles, nous faisons varier le taux de livraison des données. La taille du groupe multicast est fixée à 15 membres et la vitesse de mobilité à 1 m/sec sans temps de pause. Le fait que la vitesse de déplacement des nœuds soit relativement faible nous assure que les pertes de paquets de données sont exclusivement dues à un dépassement de tampon, une congestion ou encore des collisions dans le réseau. Pour cette expérience, la charge du trafic varie de 1 à 40 pkt/sec.

B. Résultats et analyses

Le ratio de livraison des paquets de données pour différentes charges de trafic est montré sur la figure 4.16. ODMRP offre un ratio de plus en plus faible lorsque la charge du trafic augmente. Étant donné que chaque paquet de données est inondé dans le réseau, le nombre de collisions et de dépassements de tampon augmente. De plus, de nombreux paquets de contrôle étant perdus, la construction du maillage peut être retardée et des paquets de données peuvent ne pas atteindre leurs destinataires. Malgré tout, la structure maillée permet à ODMRP de surpasser les autres protocoles multicast. En effet, pour chacun des protocoles E²M et EM2NET, le ratio de livraison diminue fortement au fur et à mesure de l'augmentation de la charge du trafic. Cette dégradation est causée par des dépassements de tampon et de nombreuses collisions. Il est même à noter que MAODV se comporte d'une meilleure façon que les deux protocoles multicast explicites pour des charges de trafic élevées. Le grand nombre de paquets de contrôle échangés dans MAODV pour maintenir la structure d'arbre lui permet en effet d'atténuer les conséquences de l'augmentation de la charge du trafic. Le taux de perte de paquets est cependant plus faible dans EM2NET que dans E²M car il offre un balancement de la charge entre les nœuds IN. Cela signifie que le trafic n'est pas concentré sur des nœuds particuliers qui pourraient constituer des points faibles de l'architecture. Dans E²M, les nœuds qui sont XF sont plus sujets à des dépassements de tampon car ils sont responsables d'un certain nombre de nœuds.

Enfin, la figure 4.17 illustre la surcharge des bytes de contrôle pour E²M et EM2NET. Nous n'avons pas jugé bon d'inclure ODMRP et MAODV dans cette expérience car la surcharge des bytes de contrôle est excessivement importante. Nous observons que EM2NET a une surcharge beaucoup plus importante pour des charges faibles (de 1 à 4 pkt/sec environ) mais il offre des performances à peu près identiques à celles de E²M lorsque les charges du trafic sont élevées. L'allure descendante des deux courbes s'explique par l'augmentation du nombre de paquets de données envoyés. En effet, plus la charge du trafic est élevée et plus le ratio entre les bytes de contrôle transmis et les bytes de données envoyés diminue.

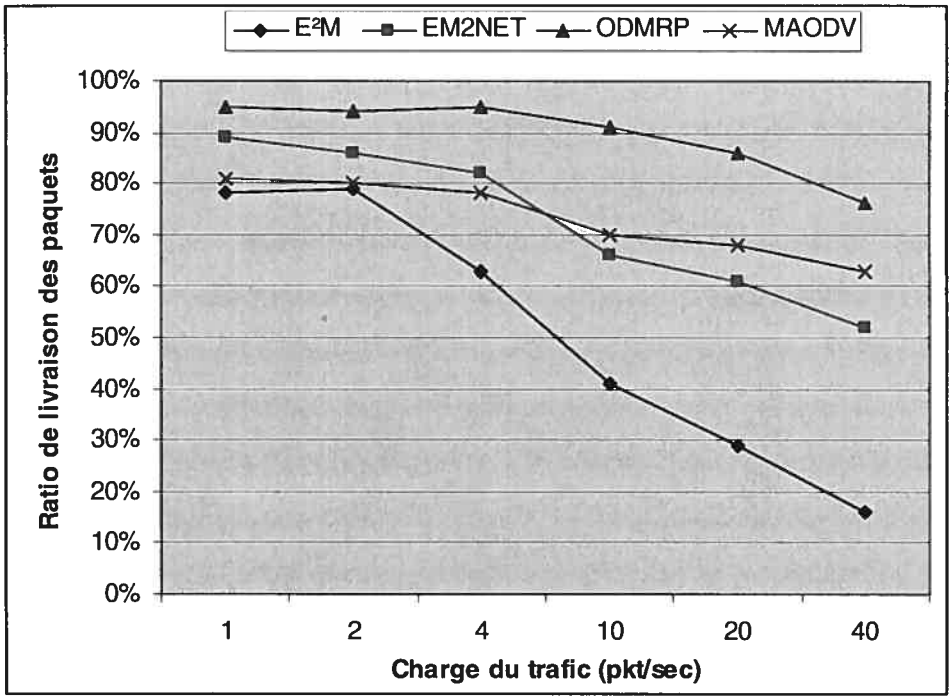

Figure 4.16. Ratio de livraison des paquets de données en fonction de la charge du trafic.

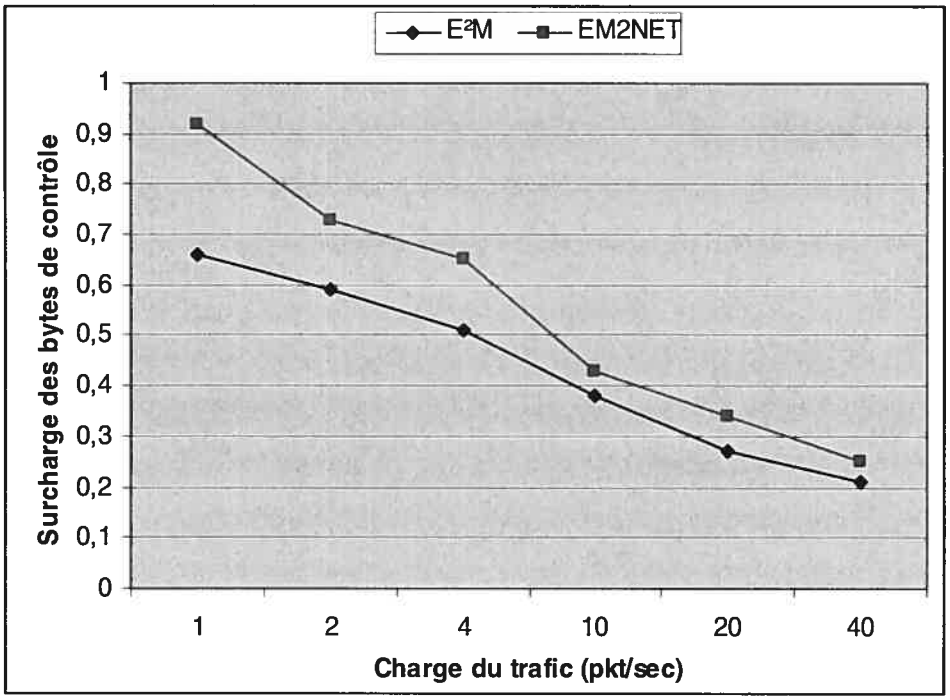

Figure 4.17. Surcharge des bytes de contrôle en fonction de la charge du trafic.

4.5 Conclusion

Les simulations nous ont permis d'appuyer les thèses que nous avancions lors de l'analyse théorique. Les résultats ainsi obtenus permettent de tirer les conclusions suivantes. Restreignons-nous dans un premier temps à E²M et EM2NET, les deux protocoles multicast explicites que nous avons choisi de comparer. Nous nous apercevons que le ratio de livraison des paquets de EM2NET est de 10 à 30% supérieur à celui de E²M. De même, la taille moyenne de l'en-tête des paquets pour EM2NET demeure relativement constante au fur et à mesure de l'augmentation de la vitesse de mobilité (aux alentours de 25 bytes) alors qu'elle croît fortement pour E²M (de 30 bytes environ jusqu'à atteindre un peu moins de 60 bytes pour une vitesse élevée). Ces deux résultats illustrent les avantages du mécanisme de réparation de l'arbre employé dans EM2NET. Comme nous l'avons dit plus haut, la gestion de la mobilité contribue à ce qu'un nœud rejoigne rapidement l'arbre lorsqu'il en est séparé, ce qui n'est pas le cas dans E²M. En effet, étant donné que la source se charge elle-même du nœud qui s'est déplacé, il peut y avoir un délai conséquent avant que celui-ci ne rejoigne l'arbre et cela peut résulter, entre autre, en des pertes de paquets.

Afin de démontrer l'efficacité de EM2NET lorsqu'il est utilisé dans des environnements ad hoc de faible densité, il était important d'étudier précisément l'influence de la taille du groupe multicast sur les performances des deux protocoles. Là encore, la procédure de réparation de l'arbre dans EM2NET contribue à un meilleur ratio de livraison des paquets. En effet, même en cas de faible mobilité, la disparition d'un nœud XF dans E²M engendre inexorablement des pertes de paquets. Cela est d'autant plus significatif lorsqu'il y a beaucoup de membres car il y a par conséquent plus de nœuds XF sélectionnés. Pour EM2NET, le ratio de livraison des paquets est proche de 90% lorsque le groupe est éparse et il est de 75% environ lorsque le groupe est dense (75 membres parmi les 75 nœuds) tandis que pour E²M, il est de 80% pour une session multicast de faible densité et il diminue fortement jusqu'à passer sous la barre des 40% pour 75 membres. En ce qui concerne la

métrique de la taille moyenne de l'en-tête, on se retrouve dans une situation similaire. À savoir, lorsque la densité du groupe est faible, EM2NET et E²M se comportent quasiment de la même façon. Cependant, la taille moyenne de l'en-tête des paquets dans E²M augmente considérablement au fur et à mesure que le groupe multicast gagne en densité. En effet, comme expliqué plus haut, la valeur du seuil pour devenir XF dans E²M pose problème tandis que le mécanisme de balancement de la charge entre les nœuds IN dans EM2NET permet de conserver une taille d'en-tête relativement faible par rapport à E²M (un nœud encode dans l'en-tête du paquet uniquement le nœud IN prédécesseur et les nœuds IN successeurs). Il s'agit d'une caractéristique particulièrement importante car cela démontre la capacité de EM2NET à mieux équilibrer la charge entre tous les nœuds du réseau et à ne pas concentrer le trafic sur des nœuds particuliers. Ces résultats se reflètent lorsque l'on quantifie la perte d'énergie des nœuds dans le réseau. Nous avons ainsi démontré qu'un nœud XF dans E²M épuisait ses batteries plus rapidement que les autres nœuds. De ce fait, lorsque la taille du groupe multicast augmente, la perte d'énergie globale des nœuds dans EM2NET est nettement inférieure à celle de E²M.

Enfin, si l'on se réfère à la charge du trafic, on s'aperçoit une fois encore que EM2NET offre un ratio de livraison des paquets plus élevé que E²M (bien que l'on observe pour les deux protocoles une forte dégradation lorsque que la charge est très élevée). Ceci s'explique de la même manière par le balancement de la charge entre les nœuds IN offert dans EM2NET.

Les simulations ont donc montré que le protocole EM2NET nouvellement proposé offrait de meilleures performances que E²M dans la majorité des scénarios utilisés. Par conséquent, il convient donc de dire qu'il est le plus performant des protocoles multicast explicites proposés à ce jour pour les réseaux ad hoc. Cependant, il est nécessaire d'analyser ses performances et de les comparer avec celles de ODMRP et de MAODV afin de montrer en quoi un protocole explicite est particulièrement adapté à un certain type de configuration : les petits groupes multicast.

Tout d'abord, comparons les deux protocoles utilisant une structure d'arbre, EM2NET et MAODV. Les simulations ont clairement montré que EM2NET offrait de meilleures performances que son homologue arborisé quelque soit le paramètre de variation (si l'on excepte des charges de trafic très élevées). Le cas standard étant fixé à un groupe multicast de 15 membres, il est donc intéressant de remarquer que EM2NET surpasse MAODV lors de l'augmentation de la vitesse de mobilité. En effet, cela montre son bon comportement lorsqu'il est utilisé dans des réseaux ad hoc de petite taille. De plus, il maintient toujours un ratio de livraison supérieur à celui de MAODV tout en consommant moins d'énergie. Ayant précédemment montré que EM2NET était le plus performant des protocoles multicast explicites et venant de prouver qu'il est le plus efficace des protocoles utilisant une structure arborescente, il s'agit maintenant d'analyser ses performances avec celles de ODMRP.

De par sa structure maillée, ODMRP n'a pas d'équivalent lorsque l'on se réfère au ratio de livraison des paquets. En effet, il offre un ratio toujours supérieur à 90% (sauf lorsque la charge du trafic est très élevée) grâce à la présence de plusieurs chemins alternatifs, ce qui signifie qu'il est aussi efficace que la méthode d'inondation. On peut donc dire qu'il est très robuste à la mobilité et n'est pas affecté par la taille du groupe multicast. Cependant, maintenir cette structure de maillage a un coût. La surcharge des bytes de contrôle est très importante dans ODMRP. Cela a une conséquence directe sur la perte d'énergie des nœuds. En effet, les nœuds du réseau dans ODMRP consomment plus d'énergie que ceux dans EM2NET. Le protocole EM2NET contribue donc à une meilleure conservation d'énergie des nœuds du réseau et la durée de vie de celui-ci s'en trouve donc accrue. Globalement, on observe que lorsque la taille du groupe multicast est faible (moins de 20 membres sur un réseau composé de 75 nœuds), EM2NET offre un ratio de livraison des paquets presque comparable à celui de ODMRP tout en échangeant beaucoup moins de paquets de contrôle, ce qui amène à une meilleure conservation d'énergie. On voit donc bien en quoi un protocole explicite étendant le schéma Xcast et par conséquent adapté aux petits groupes multicast peut être particulièrement intéressant à utiliser lorsque la configuration du réseau s'y prête.

Le tableau 4.3 suivant synthétise brièvement les résultats des simulations que nous avons menées dans ce chapitre. Soit T la taille du groupe multicast, V la vitesse de mobilité et C la charge du trafic.

Protocoles	Ratio de livraison des paquets	Taille moyenne de l'en-tête	Consommation d'énergie	Surcharge des bytes de contrôle
ODMRP	Robuste quel que soit le paramètre d'étude.	Ne s'applique pas.	Forte consommation d'énergie malgré un bon ratio.	Très supérieure à celles de E²M et EM2NET.
MAODV	Nettement inférieur à ceux de ODMRP et EM2NET.	Ne s'applique pas.	Énergie normalisée très supérieure à celle des 3 protocoles.	Très supérieure à celles de E²M et EM2NET.
EM2NET	Équivalent à celui de ODMRP lorsque T et V sont faibles.	Relativement constante quelle que soit V et T.	Meilleure conservation d'énergie que les 3 autres protocoles.	Légèrement supérieure à E²M lorsque T, V et C augmentent.
E²M	Faible ratio de livraison.	Augmente fortement avec l'augmentation de V et T.	Plus forte consommation d'énergie que EM2NET.	Offre la surcharge la plus faible des 4 protocoles.

Tableau 4.3. Synthèse des résultats de simulation.

Dans le chapitre suivant, nous allons, à partir des résultats obtenus et de la synthèse des résultats que nous venons de faire, déterminer quels sont les points à améliorer dans le protocole EM2NET et proposer des solutions afin d'améliorer ses performances. En effet, certains scénarios de simulation ont mis en avant les carences du protocole. En particulier, nous avons remarqué que EM2NET a une surcharge des bytes de contrôle plus élevée que celle de E²M lorsque l'environnement est soumis à une forte mobilité, lorsque la taille du groupe multicast augmente ou encore lorsque la charge du trafic s'intensifie. Bien que nous ayons

montré que ces paquets de contrôle sont importants car ils servent à maintenir la connectivité du groupe, il serait intéressant de diminuer le nombre de leurs échanges périodiques entre les nœuds IN. De plus, minimiser la perte d'énergie des nœuds dans le réseau est un objectif particulièrement important lorsque les nœuds évoluent dans un environnement ad hoc. Il serait donc judicieux de proposer un mécanisme permettant de restreindre la consommation d'énergie à son strict minimum. Puis nous présentons dans une seconde partie une méthode hybride basée sur EM2NET et ODMRP. À partir des observations que nous venons de faire, nous proposons un protocole fonctionnant en alternance suivant le mode d'utilisation de EM2NET et ODMRP. Il permet ainsi de s'adapter à tout type d'environnement et de garantir les meilleures performances quelque soient les paramètres du réseau.

Chapitre 5

Solutions pour améliorer le protocole EM2NET

Le chapitre précédent nous a permis de démontrer l'utilité d'un protocole multicast explicite lorsqu'un réseau ad hoc est constitué d'un grand nombre de petits groupes. Nous avons montré durant les simulations que EM2NET était le plus efficace des protocoles de sa catégorie (comparativement à E²M pour les protocoles explicites et MAODV pour les protocoles basés sur une structure d'arbre) et qu'il obtenait même de meilleures performances que ODMRP dans certaines situations. En particulier, il offre une meilleure conservation d'énergie ainsi qu'une surcharge des bytes de contrôle moindre. Cependant, nous avons également pu identifier les points faibles de son architecture. Dans ce chapitre, nous allons détailler ces points faibles et proposer des solutions et/ou des modifications afin de le rendre plus robuste. Puis dans la dernière partie, nous présentons un protocole hybride basé sur une alternance entre les protocoles ODMRP et EM2NET après amélioration.

5.1 Améliorations et modifications de EM2NET

5.1.1 Diminution du nombre de messages de contrôle périodiques

La clé du succès du protocole EM2NET est l'échange périodique de messages de contrôle entre les nœuds IN. En effet, cela permet de maintenir efficacement la structure de l'arbre multicast. Chaque nœud qui est un membre de l'arbre ou un

nœud de branchement devient un nœud IN et est par la suite responsable de l'envoi de messages BRANCH_UP et BRANCH_DOWN respectivement vers son nœud IN prédécesseur et vers ses nœuds IN successeurs. Cependant, cet échange périodique entraîne une surcharge en terme de bytes non négligeable. C'est ce que nous avons montré durant l'analyse théorique et les simulations (Cf. chapitre 4). Un nœud, lors de la réception d'un message de contrôle, consomme une certaine quantité d'énergie nécessaire au traitement du message et à son éventuelle retransmission. Il est donc primordial de minimiser le plus possible l'envoi des messages de contrôle afin que les nœuds ne puisent pas inutilement dans leur réserve d'énergie tout en conservant le bénéfice engendré par l'utilisation de tels messages. Nous présentons dans la section ci-après une solution possible à ce problème.

A. Solution proposée

Afin d'éviter la propagation d'un trop grand nombre de messages de contrôle, une alternative pourrait être de considérer les messages de données comme des messages BRANCH_DOWN implicites. Ainsi, lorsqu'un nœud reçoit un paquet de données, il peut en déduire que le nœud IN prédécesseur fait toujours partie de l'arbre. Il convient alors d'utiliser un temporisateur adéquat (appelons-le A), c'est-à-dire un temporisateur inférieur au temporisateur utilisé pour la gestion de la mobilité et de la réparation des routes (appelons-le B) : lorsqu'un nœud reçoit un message de type DATA avant l'expiration de A, il réinitialise les temporisateurs A et B ; si aucun message de données n'est reçu avant l'expiration de A, le nœud IN se charge alors d'envoyer, et uniquement dans ce cas, un message BRANCH_DOWN aux nœuds IN successeurs (avant l'expiration de B afin que les nœuds successeurs sachent que le nœud IN prédécesseur est toujours dans l'arbre). Il est à prévoir que cette méthode sera efficace quelle que soit la charge du trafic car il y aura, quoiqu'il en soit, très peu de transmissions de messages BRANCH_DOWN. Nous pouvons même avancer que ce nombre sera nul lorsque la charge du trafic sera élevée (car la probabilité qu'un paquet de donnée parvienne à sa destination est plus grande). Les

messages BRANCH_DOWN ne sont ainsi conservés que pour des cas particuliers : lorsque le flot de données est stoppé ou lorsque les messages sont perdus plusieurs fois consécutivement (non réception de messages BRANCH_UP de la part des nœuds IN successeurs).

B. Analyse théorique

Dans un premier temps, nous allons mener une analyse théorique comparable à celle du chapitre 4 afin de montrer les avantages de cette méthode. Nous rappelons ci-dessous les deux formules permettant de calculer le nombre n de messages de contrôle périodiques échangés au cours d'une instance de simulation, où x_1 est la taille du groupe multicast, x_2 le nombre de nœuds XF dans E²M, d le nombre de nœuds IN successeurs dans EM2NET (soit le nombre de messages BRANCH_DOWN transmis), t_3 la durée de la simulation, t_1 le taux de livraison des messages BRANCH et t_2 le taux de livraison des messages de contrôle dans E²M :

$$\text{(EM2NET)} \quad n = [x_1 \times (1+d)] \times (t_3 / t_1)$$

$$\text{(E²M)} \quad n = [x_1 - (x_2 \times 8) + x_2] \times (t_3 / t_2)$$

Si l'on suppose qu'il n'y a pas de perte de paquets, une charge de trafic de 2 pkt/sec nous permet d'affirmer qu'aucune transmission de messages BRANCH_DOWN ne sera effectuée (d est égal à 0) étant donné que l'intervalle d'envoi de ces messages est de 5 secondes. D'après le tableau 4.2, nous observons que pour une session multicast de 5 membres, un seul nœud XF est sélectionné parmi l'ensemble des nœuds tandis qu'il y en a 6 lorsque le réseau compte 75 membres. Les résultats sont alors les suivants. Pour un groupe multicast de 5 membres, la valeur de n est estimée à 600 paquets de contrôle pour EM2NET et 1000 pour E²M. Lorsqu'il y a 75 membres dans le réseau, il y a 9000 et 6600 paquets de contrôle

échangés respectivement pour EM2NET et E²M. Ces valeurs ne changent pas en ce qui concerne E²M mais on s'aperçoit que la diminution est exactement de 50 % pour le protocole EM2NET par rapport à sa version originale (n était alors compris entre 1200 et 18 000). En effet, il n'y a pas de messages BRANCH_DOWN envoyés, ce qui explique un tel gain. Toutefois, ces résultats sont à prendre avec précaution car rappelons-le, les pertes de paquets ne sont pas prises en compte. Nous allons voir dans la section suivante ce que donnent réellement les simulations.

C. Résultats des simulations

Afin de montrer l'efficacité de la solution proposée, plaçons-nous dans un environnement de simulation comparable à celui du chapitre 4. Le cas standard est le suivant : la taille du groupe multicast est fixée à 15 membres, la vitesse de mobilité à 1 m/sec sans temps de pause et la charge du trafic à 2 pkt/sec. Les courbes qui vont suivre font le parallèle entre la version originale du protocole EM2NET et la version implémentant les modifications. De plus, dans un souci de lisibilité, nous n'avons pas inclus les protocoles MAODV et ODMRP dans les courbes lors du calcul de la surcharge des bytes de contrôle car les précédentes expériences ont montré qu'ils étaient tous les deux très peu performants pour cette métrique.

Étudions tout d'abord l'influence de la charge du trafic sur la surcharge des bytes de contrôle. Nous faisons varier le débit des paquets de données de 1 à 40 pkt/sec tandis que la vitesse de mobilité et la taille du groupe multicast sont standardisées. La figure 5.1 illustre les résultats de cette étude. Nous pouvons remarquer que la solution proposée offre une diminution conséquente du nombre de messages de contrôle échangés entre les nœuds IN. Plus précisément, pour des charges de trafic faibles (de 1 à 4 pkt/sec), le gain est d'environ 10 % par rapport à la version originale de EM2NET. Lorsque la charge du trafic s'intensifie, le gain devient considérable (jusqu'à 50 %) et la surcharge des bytes de contrôle est alors bien inférieure à celle de E²M. Cela s'explique par le fait que les nombreux paquets de données délivrés jouent le rôle de paquets de contrôle implicites (il n'y a

quasiment plus aucune transmission de messages BRANCH_DOWN) et contribuent ainsi à une surcharge très faible. Il est intéressant de remarquer que pour des charges de trafic élevées, la diminution est comparable au cas idéal dont nous avons parlé durant l'analyse théorique. En effet, un grand nombre de paquets de données sont transmis ce qui augmente les chances de livraison avant l'expiration du temporisateur associé à la transmission des messages de contrôle (et ce, même si des pertes de paquets surviennent). C'est la raison pour laquelle le gain avoisine les 50 % tandis qu'il n'est que de 10 % pour des charges plus faibles.

Faisons à présent varier la taille du groupe multicast de 5 à 75 membres (les autres paramètres sont normalisés selon le cas standard). Notons que le taux de livraison des données est volontairement faible afin de placer les protocoles sur un pied d'égalité. Le nombre de bytes de contrôle transmis par rapport au nombre de bytes de données envoyés est présenté sur la figure 5.2. Encore une fois, on s'aperçoit que la méthode de diminution des messages de contrôle porte ses fruits : le gain est de 40 % environ pour une taille de groupe multicast supérieure à 30 membres. Lorsque le groupe est éparse (moins de 30 membres), il y a par définition moins de nœuds IN et donc moins d'échanges de paquets de contrôle, ce qui explique que le gain n'est pas autant significatif (de 5 à 20 %). Au contraire, nous observons que lorsqu'il y a plus de membres dans la session multicast, la surcharge des bytes de contrôle diminue comparativement à la version originale de EM2NET. Bien sûr, le protocole E²M conserve un avantage certain du à la formation de nœuds XF dans le réseau, ce qui contribue à une meilleure surcharge des bytes de contrôle au fur et à mesure que le groupe multicast se densifie. Il convient toutefois de rappeler que les protocoles multicast explicites que sont E²M et EM2NET sont avant tout conçus pour supporter des groupes multicast de petite taille. Si l'on considère un groupe de 15 membres, nous observons d'après les deux courbes que nous venons de présenter que EM2NET offre une diminution du nombre de paquets de contrôle périodiques et que cette diminution est d'autant plus avantageuse lorsque la charge du trafic s'intensifie. Nous pouvons en conclure que EM2NET est parfaitement adapté aux environnements ad hoc pour lesquels il est développé.

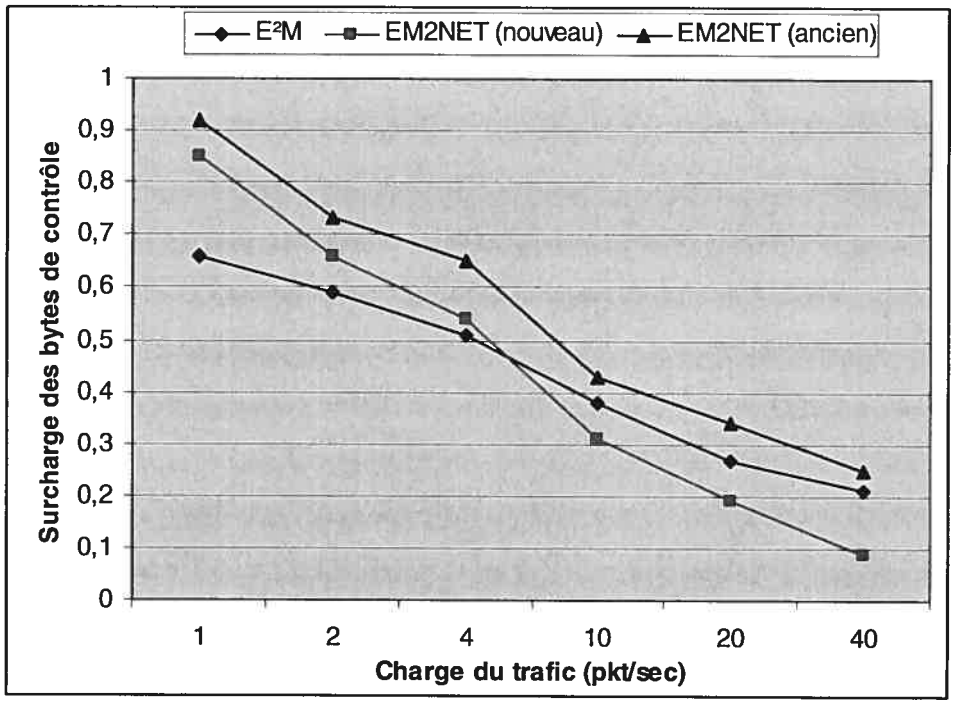

Figure 5.1. Surcharge des bytes de contrôle en fonction de la charge du trafic.

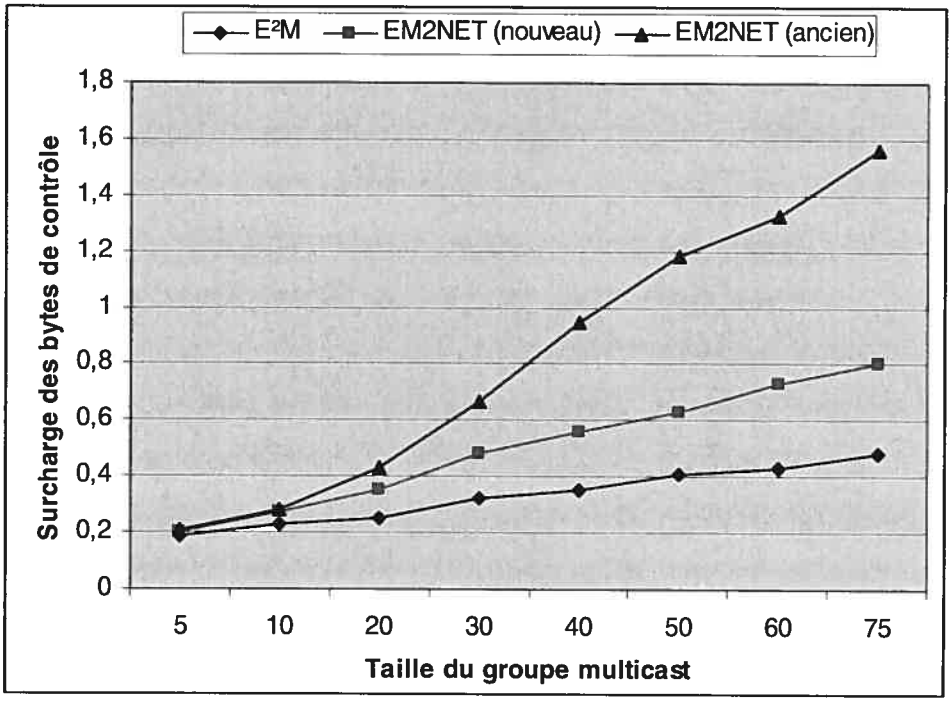

Figure 5.2. Surcharge des bytes de contrôle en fonction de la taille du groupe multicast.

Nous venons de proposer une méthode permettant de réduire le nombre de messages de contrôle échangés au sein du réseau entre les nœuds IN. Il s'agit donc maintenant d'analyser le gain d'énergie consécutif à cette diminution. La figure 5.3 est une illustration de la consommation d'énergie des nœuds en fonction de la charge du trafic tandis que la figure 5.4 en étudie les effets sur le ratio de livraison de paquets.

EM2NET était, d'après les précédentes analyses, le plus performant des quatre protocoles : l'énergie normalisée (soit l'énergie moyenne consommée par paquet de données) était, quelque soit le paramètre d'étude, en dessous de celle des autres protocoles. C'est pour cette raison que nous avons choisi, toujours dans un souci de clarté, de comparer uniquement les différentes versions du protocole EM2NET et de faire abstraction de ODMRP, MAODV et E²M (Cf. par exemple figure 4.12). Conformément à cela, nous nous apercevons que la version de EM2NET implémentant les modifications obtient de meilleures performances que sa version originale lorsque l'on étudie la variation de la charge du trafic. Ces résultats se justifient bien sûr par l'efficacité de la méthode mais ce n'est pas la seule explication. En effet, il y a moins de paquets circulant au sein du réseau (et ce malgré l'augmentation de la charge) et donc moins de possibles collisions et pertes de paquets. Le ratio de livraison des données est donc meilleur. La figure 5.4 confirme cette idée. La perte d'énergie des nœuds augmente alors mais pas aussi sévèrement que dans la version originale. Cela explique l'allure constante de la courbe et ainsi l'amélioration constatée sur l'énergie normalisée.

Focalisons-nous à présent sur l'énergie normalisée en fonction de la taille du groupe multicast. Les résultats sont exposés sur la figure 5.5. Nous remarquons encore une fois que la méthode proposée permet une amélioration significative de la consommation d'énergie des nœuds. De la même façon que pour la courbe précédente, le ratio de livraison des paquets est en légère amélioration et les nœuds épuisent moins rapidement leurs batteries. Il s'en suit une nette augmentation de la durée de vie du réseau.

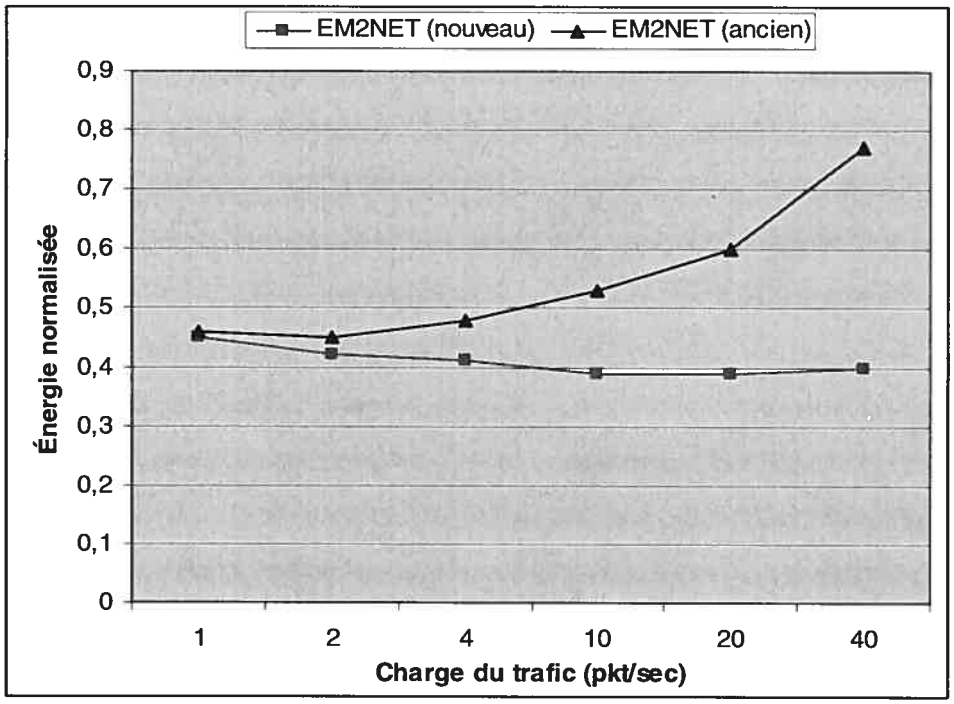

Figure 5.3. Énergie normalisée en fonction de la charge du trafic.

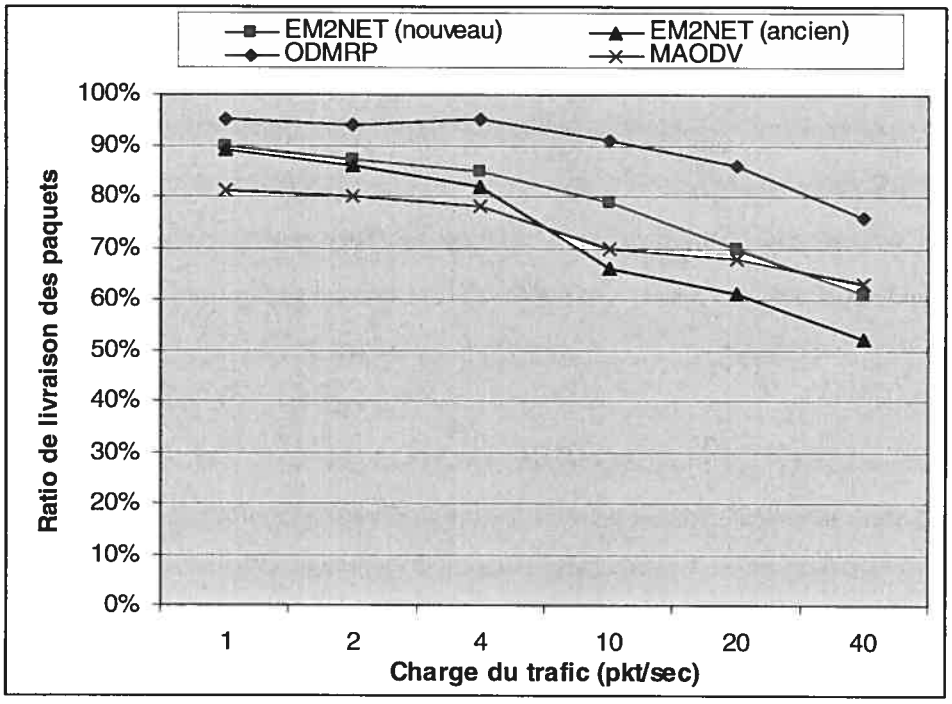

Figure 5.4. Ratio de livraison des paquets de données en fonction de la charge du trafic.

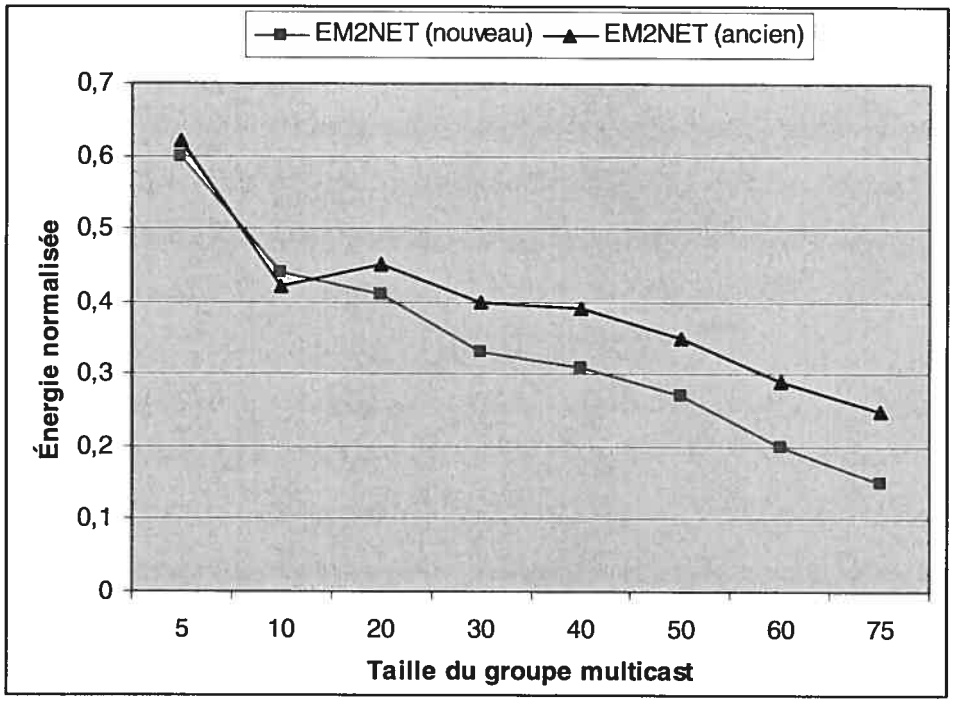

Figure 5.5. Énergie normalisée en fonction de la taille du groupe multicast.

5.1.2 Une nouvelle méthode de jointure au groupe multicast

Il s'agit encore une fois de proposer une méthode permettant de réduire le nombre de messages de contrôle propagés au sein du réseau. En effet, dans la section précédente, nous venons de prouver que la diminution de la surcharge des bytes de contrôle avait une conséquence directe sur la consommation d'énergie des nœuds. Tous les moyens sont donc bons pour minimiser le plus possible cette surcharge. Lors de la description du protocole EM2NET, nous avons indiqué qu'un nœud souhaitant joindre un groupe multicast le faisait par l'intermédiaire d'un message JOIN envoyé en unicast vers la source. Lors de la réception de ce message, la source y répond par un message BRANCH_ACK confirmant l'appartenance du nouveau nœud au groupe. L'idée est de présenter une méthode de jointure à la session multicast basée sur une recherche locale permettant ainsi de s'abstenir des messages BRANCH_ACK propagés de nœuds en nœuds jusqu'au nouveau membre.

A. Solution proposée

Lorsqu'un nœud désire joindre le groupe multicast, il diffuse un message JOIN avec un TTL limité afin de repérer localement un nœud IN dans l'arbre multicast. S'il y parvient, le nœud IN met à jour sa table MFIT (le nouveau nœud fait maintenant partie de sa liste des nœuds IN successeurs) et envoie un message BRANCH_DOWN afin d'avertir le nouveau nœud qu'il fait désormais partie du groupe. Il est également possible que le nœud trouvé devienne un nœud de branchement (c'est à lui de faire la vérification). Lors de la réception de ce message, le nouveau nœud se considère comme un nœud IN, initialise sa table MFIT avec comme nœud IN prédécesseur le nœud IN trouvé et peut également déterminer s'il est un nœud de branchement grâce à la liste de tous les nœuds IN successeurs contenus dans le message BRANCH_DOWN. Si la diffusion échoue, l'ancienne méthode est utilisée : envoi d'un message JOIN directement vers la source.

Ce processus évite d'une part l'envoi d'un message unicast de type JOIN dirigé vers la source mais surtout, cela permet de ne pas avoir à envoyer en réponse un message BRANCH_ACK jusqu'au nouveau membre, d'où une grande économie de messages de contrôle puisque les messages BRANCH_ACK sont propagés par chaque nœud jusqu'au nouveau membre. Toutefois, elle ne s'avère efficace que lorsque des nœuds sont susceptibles de joindre le groupe multicast au cours de la simulation. Si tous les nœuds joignent le groupe au début de la session, cette technique n'est d'aucune efficacité.

Dans ce cas (lorsque plusieurs nœuds joignent le groupe multicast au début de la simulation), la méthode originale est utilisée. Cependant, il est tout de même intéressant de limiter le nombre de messages BRANCH_ACK envoyés en réponse aux messages JOIN. En effet, dans le cas normal, il y a un message BRANCH_ACK transmis par nouveau nœud désirant joindre le groupe. L'idée est alors d'attendre un certain temps t afin de recevoir tout ou partie des messages JOIN puis de propager un et un seul message BRANCH_ACK avec la liste de tous les nouveaux membres.

B. Résultats des simulations

Plaçons-nous dans la situation suivante : un réseau de 75 nœuds se déplaçant à une vitesse de 1 m/sec sans temps de pause et une charge de trafic de 2 pkt/sec. Afin d'identifier clairement la pertinence de la méthode proposée, supposons que les nœuds sont susceptibles de joindre et/ou de quitter le groupe multicast à tout moment. Nous allons donc appliquer la solution aux opérations de jointure mais également aux opérations de retrait du groupe (messages LEAVE). Pour cela, nous avons choisi de borner supérieurement chaque instance de simulation à un nombre m de membres. Plus précisément, le nombre de nœuds dans le groupe multicast va varier au cours de la simulation de 5 à m (5 nœuds joignent la session au début de la simulation et le nombre de membres augmente ou diminue jusqu'à atteindre m). Nous faisons varier les bornes supérieures de 10 à 75 membres et comparons E²M avec la version originale de EM2NET (version 1 ou V1), la version implémentant la méthode proposée dans la section 5.1.1 (version 2 ou V2) et la version implémentant les deux nouvelles méthodes (version 3 ou V3). Les résultats relatifs à la surcharge des bytes de contrôle sont indiqués sur la figure 5.6. Rappelons que les paquets de contrôle sont les messages BRANCH périodiques mais aussi et surtout les messages JOIN et LEAVE et les réponses BRANCH_ACK correspondantes. Nous remarquons que la version 2 de EM2NET diminue effectivement le nombre de paquets de contrôle échangés au sein du réseau (car elle implémentation la première méthode) mais nous savons que cela concerne uniquement les messages BRANCH_DOWN. En effet, la configuration de simulation que nous avons choisi d'adopter résulte en un grand nombre de messages JOIN et LEAVE adressés à la source. Les paquets relatifs aux opérations de jointure et de retrait du groupe multicast sont alors nombreux, d'où l'augmentation conséquente de la surcharge. Au contraire, la version 3 de EM2NET présente une surcharge très faible (relativement proche de celle de E²M) car les solutions implémentées permettent d'une part de diminuer le nombre de paquets de contrôle entre les nœuds IN et d'autre part d'utiliser une procédure locale lorsqu'un nœud désire joindre ou quitter le groupe. Malgré le fait que la méthode échoue de

temps à autre (s'il n'y a pas de nœuds IN dans la zone de recherche délimitée par le TTL), le gain est tout de même important par rapport aux autres versions.

Identiquement à la solution présentée précédemment, il est intéressant d'analyser le gain obtenu en terme de consommation d'énergie lors de l'utilisation de cette méthode. Là encore, nous analysons les performances des protocoles E²M et EM2NET (sous ses 3 versions). Les données de simulation présentées sur la figure 5.7 confirment les résultats antérieurs. L'énergie normalisée de E²M est toujours très supérieure à celle de EM2NET, quelque soit la version utilisée. Si l'on compare maintenant les trois implémentations de EM2NET, on s'aperçoit qu'effectivement, la version qui cumule les deux méthodes que nous venons de présenter s'en sort avec une énergie normalisée très en baisse par rapport aux deux autres. En effet, l'allure fortement descendante de la courbe s'explique par une consommation d'énergie faible comparativement au ratio de livraison des données.

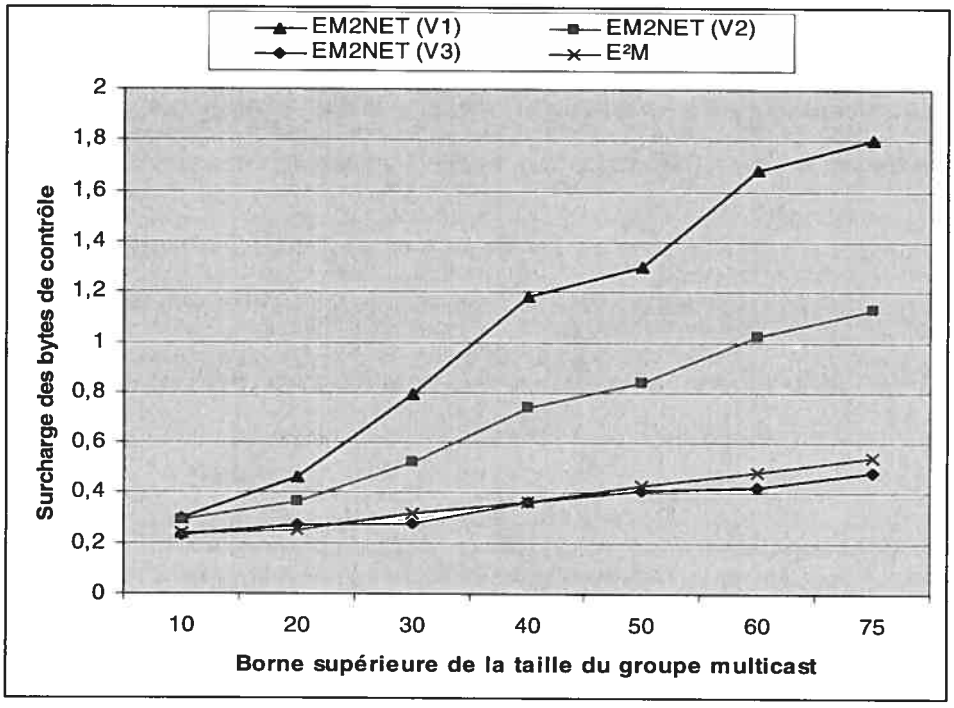

Figure 5.6. Surcharge des bytes de contrôle en fonction de la borne supérieure de la taille du groupe multicast.

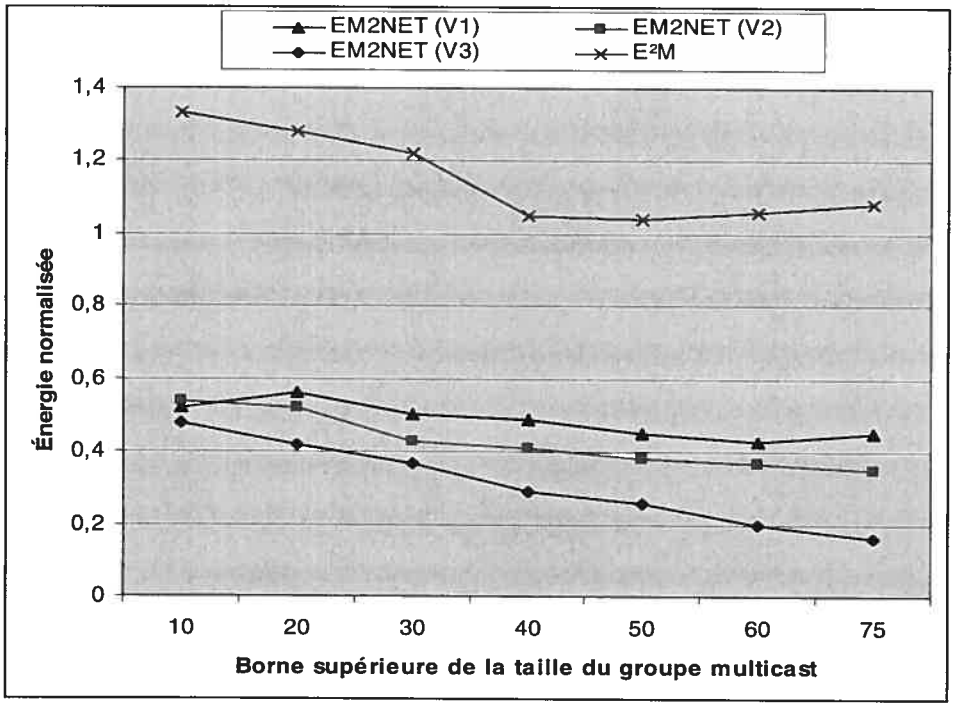

Figure 5.7. Énergie normalisée en fonction de la borne supérieure de la taille du groupe multicast.

5.2 Une solution hybride au multicast dans MANETs

Les simulations que nous avons mené dans le chapitre 4 ainsi que dans la première partie du présent chapitre nous ont permis de mettre en avant les avantages et les inconvénients de chacun des protocoles que nous avons comparé (E²M, EM2NET, ODMRP et MAODV). En particulier, nous avons pu identifier dans quelles situations tel protocole se comportait d'une meilleure façon que tel autre. Par exemple, nous avons remarqué que le protocole EM2NET obtenait de bons résultats lorsque la taille du groupe multicast était faible (ce qui était le but recherché étant donné que EM2NET est avant tout destiné aux petits groupes multicast) tandis que ODMRP était particulièrement performant lorsque la session multicast gagnait en densité. Conformément à ces observations, nous proposons dans cette section un protocole hybride qui repose sur l'utilisation en alternance des protocoles EM2NET

et ODMRP. Un protocole hybride est un protocole qui combine généralement deux ou plusieurs techniques afin de gagner en robustesse et en performance [36]. L'idée générale est la suivante : il s'agit d'analyser périodiquement au cours de la simulation certains paramètres du réseau (plus particulièrement le nombre de membres dans le groupe multicast, la vitesse de mobilité et la charge du trafic) et de prendre ou non la décision de changer le mode de fonctionnement du protocole. Nous allons dans un premier temps définir avec précision quelles sont les situations qui peuvent engendrer un basculement entre les modes EM2NET et ODMRP (dépassement d'un seuil par exemple). Puis nous détaillons les changements nécessaires à l'implémentation du protocole hybride. En effet, les deux protocoles que sont EM2NET et ODMRP sont basés sur une structure différente (respectivement un arbre et un maillage). Nous allons donc proposer une méthode permettant de passer efficacement d'un mode à l'autre. Le but de ce protocole est de s'adapter dynamiquement à l'environnement du réseau afin d'offrir, en tout temps, les meilleures performances (en termes de consommation d'énergie, de ratio de livraison des paquets de données et de surcharge des bytes de contrôle).

5.2.1 Définition des règles de basculement

Commençons tout d'abord par étudier l'influence de la taille du groupe multicast sur les performances de EM2NET et ODMRP. D'après les résultats des simulations du chapitre 4 (Cf. section 4.4.1), nous remarquons que les deux protocoles offrent des ratios de livraison de paquets sensiblement équivalents pour une taille de groupe inférieure à 20 membres (seulement 5 % de différence en faveur de ODMRP). Au-delà de cette valeur, ODMRP est plus efficace que EM2NET et ses performances vont même en s'améliorant au fur et à mesure de l'augmentation du nombre de membres dans la session. En contrepartie, EM2NET présente une surcharge en terme de bytes de contrôle très faible par rapport à ODMRP (quelle que soit la taille du groupe) et cela est d'autant plus vrai lorsque l'on implémente les

méthodes proposées dans la section 5.1 du chapitre 5. De plus, si l'on étudie avec précision la figure 4.12, nous nous apercevons que l'énergie normalisée de ODMRP est très nettement supérieure à celle de EM2NET lorsque le groupe multicast est composé de moins de 20 membres. Il convient ainsi de dire qu'une valeur de seuil de 20 nœuds pour la taille du groupe multicast (sur un ensemble de 75 nœuds) apparaît comme un très bon compromis ratio de livraison/consommation d'énergie pour le basculement entre les modes EM2NET et ODMRP.

Il s'agit maintenant d'analyser les caractéristiques des protocoles EM2NET et ODMRP sous différents scénarios de mobilité. En ce qui concerne le ratio de livraison des données, nous constatons que lorsque la vitesse de mobilité est inférieure à 5 m/sec, EM2NET et ODMRP offrent des ratios relativement semblables. Les pertes de paquets dans le protocole EM2NET sont certes légèrement plus importantes mais il convient d'analyser parallèlement à cela l'énergie normalisée des nœuds. En effet, la figure 4.15 nous montre que ODMRP est beaucoup moins performant que EM2NET en terme de consommation d'énergie mais que cette différence tend à diminuer lorsque la vitesse de déplacement des nœuds augmente. En se basant sur ces observations, nous établissons le scénario de basculement suivant. Lorsque les nœuds se déplacent à une vitesse inférieure à 5 m/sec, le protocole EM2NET est utilisé. Si la mobilité augmente (entre 5 m/sec et jusqu'à 10 m/sec ou plus), le protocole hybride bascule alors en mode ODMRP.

Enfin, nous allons déterminer la valeur du taux de livraison des paquets qui engendre un basculement entre les protocoles EM2NET et ODMRP. La section 4.4.3 du chapitre 4 indique qu'au dessus de 4 pkt/sec, le ratio de livraison des paquets de données pour EM2NET décroît rapidement. Nous avons justifié cette chute par les nombreuses collisions et dépassements de tampons. Cependant, nous avons montré que la solution proposée dans la section 5.1.1 du chapitre 5, qui vise à diminuer le nombre de messages de contrôle échangés au sein du réseau, permettait non seulement de réduire la consommation d'énergie des nœuds mais aussi et surtout d'offrir un ratio de livraison meilleur que lors de l'exécution de la version originale

(Cf. figure 5.4). Ainsi, il convient d'ajuster le seuil à une valeur de 10 pkt/sec. En dessous d'une telle charge, EM2NET offre une meilleure énergie normalisée que ODMRP couplée à un bon taux de livraison de paquets tandis que pour une charge de trafic supérieure, le gain en énergie n'est pas suffisamment important pour que l'on néglige les pertes de paquets (plus de 15 % de différence pour des charges supérieures à 10 pkt/sec).

Nous venons de définir numériquement les seuils entraînant un changement dans le mode de fonctionnement du protocole hybride (EM2NET ou ODMRP). Nous nous sommes efforcés d'établir des valeurs qui lui permettent, en théorie, de consommer le moins d'énergie possible tout en maintenant un bon ratio de livraison de paquets et une surcharge des bytes de contrôle la plus faible possible. Le tableau 5.1 suivant fait la synthèse de ces résultats. Soit T la taille du groupe multicast, V la vitesse de mobilité et C la charge du trafic. Il convient de rappeler que ces valeurs ont été définies sur la base d'un réseau ad hoc composé de 75 nœuds. Si la taille du réseau venait à varier, les valeurs de seuils pourraient être calculées au prorata des valeurs indiquées dans le tableau. Dans le paragraphe suivant, nous présentons une méthode pour basculer, conformément aux règles ainsi définies, d'un protocole à l'autre puis nous donnons les détails de l'implémentation du protocole hybride sous le logiciel de simulation ns-2.

Paramètres du réseau	Mode de fonctionnement
T < 20 membres	EM2NET
T > 20 membres	ODMRP
V < 5 m/sec	EM2NET
V > 5 m/sec	ODMRP
C < 10 pkt/sec	EM2NET
C > 10 pkt/sec	ODMRP

Tableau 5.1. Règles de basculement entre les modes EM2NET et ODMRP.

5.2.2 Implémentation du protocole hybride

Nous avons rappelé précédemment que les protocoles EM2NET et ODMRP reposent sur une structure complètement différente. ODMRP est basé sur une structure maillée (Cf. section 3.2.1 du chapitre 3) et plusieurs routes alternatives existent ainsi entre deux nœuds. De son côté, EM2NET établit et maintient une structure d'arbre, ce qui signifie qu'à un instant donné, un et un seul chemin est disponible pour la livraison des données (Cf. section 4.1 du chapitre 4). De plus, l'arbre construit par EM2NET est indépendant du maillage de ODMRP. En effet, à la différence d'un protocole hybride tel que AMRoute où la structure d'arbre est établie à partir de la structure maillée, la probabilité que le chemin employé dans EM2NET ne soit pas inclus dans le maillage de ODMRP est très forte. Le défi majeur de l'implémentation du protocole hybride consiste donc à proposer une méthode permettant de basculer efficacement d'une structure à l'autre sans affecter ses performances. En particulier, il est important que le changement de mode de fonctionnement entraîne dans la mesure du possible très peu de pertes de paquets, ce qui aurait un impact certain sur le ratio de livraison des données. Les bénéfices recherchés par l'utilisation d'un protocole hybride seraient alors perdus. De plus, la technique de basculement doit permettre de minimiser le nombre de paquets de contrôle nécessaires à l'établissement d'une nouvelle structure. Nous allons présenter dans la section suivante une méthode qui répond à tous ces critères.

A. Solution proposée

Il est tout d'abord nécessaire de donner des précisions quand aux règles de basculement définies précédemment. En effet, durant l'exécution du protocole, il peut y avoir des situations ambiguës. Considérons par exemple T inférieur à 20 membres et C supérieur à 10 pkt/sec (Cf. tableau 5.1). Dans ce cas, quel mode de fonctionnement choisir ? EM2NET ou ODMRP ? Afin de clarifier la situation, nous

allons adopter un principe de pondération lors de l'évaluation des paramètres de la simulation. Plus précisément, nous affectons un poids à chacune des trois variables dans le but d'établir une hiérarchie pour la décision de basculement. Cette hiérarchie est la suivante, par ordre de priorité : T, V et C. Nous avons choisi T (la taille du groupe multicast) comme étant le seuil prioritaire dans le choix du protocole à utiliser car EM2NET est un protocole explicite spécialement conçu pour les sessions multicast de petite taille et il est donc judicieux de l'utiliser lorsque l'environnement s'y prête, même si les autres paramètres lui sont défavorables. En effet, les simulations ont montré que les résultats les plus favorables à EM2NET (aussi bien en terme de ratio de livraison de paquets que d'énergie normalisée) étaient obtenus lorsque la taille du groupe multicast était faible et ce, indépendamment des scénarios de mobilité et des différentes charges de trafic.

Ainsi, nous analysons seconde par seconde la valeur de T. Lorsque celle-ci se situe en dehors de la zone de 20 membres à ± 5 membres (soit entre 15 et 25 membres), le comportement du protocole hybride est dicté par T. Par exemple, si T est égal à 30, le protocole hybride fonctionne en mode ODMRP quelles que soient les valeurs de V et C. À l'inverse, si la valeur de T approche le seuil de 20 membres (à ± 5 membres), nous prenons la décision de basculer le mode de fonctionnement suivant la valeur de V. Enfin, et de la même façon, si la valeur de V est proche du seuil de 5 m/sec (à ± 1 m/sec), c'est la valeur de C qui déterminera si le basculement doit avoir lieu ou pas.

Si la décision de basculement est imminente (c'est-à-dire que l'on approche un des trois seuils), il s'agit alors de mettre en place une nouvelle structure pour la livraison des données en prévision du changement de mode de fonctionnement. En effet, il est important que la structure ait été établie correctement lorsque le basculement sera effectif afin de bénéficier le plus vite possible des performances de l'un ou l'autre des protocoles (cette technique est aussi connue sous le nom de *make and break*). Le fonctionnement standard de EM2NET ou ODMRP est alors adopté.

Dans le cas ou le basculement n'intervient pas ou si un nouveau changement du mode de fonctionnement intervient rapidement, la structure préalablement établie est conservée en cache durant un laps de temps fixé à 2 secondes afin de ne pas avoir à établir une nouvelle structure si le basculement a finalement lieu ou si un autre changement intervient. Ce mécanisme de cache est déployé à des fins d'économie de paquets de contrôle. Nous avons choisi cette valeur de 2 secondes pour le cache car une plus grande valeur engendrerait trop de changements dans la nouvelle structure (de nombreux membres pourraient avoir joint ou quitté la session entre temps).

B. Implémentation

Intéressons-nous à présent à l'implémentation proprement dite du protocole hybride sous le logiciel de simulation ns-2. Nous disposons des agents ODMRP et EM2NET (nous utilisons la version de EM2NET mettant en œuvre les modifications apportées dans la section 5.1 du chapitre 5). Il s'agit donc d'implémenter les règles de basculement que nous avons définies précédemment. Cela se fait à l'intérieur du script OTcl qui, rappelons-le, est le script définissant la topologie du réseau et les paramètres de la simulation. Lorsque l'analyse périodique des différentes variables du réseau conclut à un basculement imminent du mode de fonctionnement du protocole hybride (Cf. paragraphe précédent), il faut tout d'abord établir la nouvelle structure puis il s'agit d'indiquer aux nœuds membres qu'ils doivent soit adopter le fonctionnement de l'agent EM2NET soit celui de l'agent ODMRP (conformément à la décision prise). En ce qui concerne EM2NET, les membres du groupe multicast doivent initier la procédure de jointure que nous avons définie dans la section 5.1.2 du chapitre 5 tandis que dans le cas de ODMRP, une phase de requête/réponse est nécessaire à l'établissement de la structure maillée.

C. Résultats

Afin de montrer clairement l'efficacité du protocole hybride, nous allons comparer ses performances avec celles de ODMRP et EM2NET (version avec

modifications). Les métriques de comparaison sont identiques à celles du chapitre 4 : ratio de livraison des paquets de données, surcharge des bytes de contrôle et énergie normalisée. De même, nous faisons varier trois paramètres afin d'altérer les résultats des simulations mais également afin de mettre en œuvre explicitement un basculement entre les modes EM2NET et ODMRP (Cf. section 4.2.2 du chapitre 4). En effet, à la différence du chapitre 4 où les courbes résultaient d'une moyenne de plusieurs instances de simulation, nous varions dans cette section les paramètres au cours de la même instance de simulation afin qu'au moins un basculement ait lieu.

La figure 5.8 est une illustration du ratio de livraison des paquets en fonction de la taille du groupe multicast. La vitesse de déplacement des nœuds est fixée à 5 m/sec. Nous remarquons qu'au début de la simulation, les performances du protocole hybride sont identiques à celles de EM2NET car ce dernier est à cet instant le protocole en exécution. Lorsque la taille du groupe atteint le seuil de 20 membres, un basculement a lieu. En effet, la charge du trafic (2 pkt/sec) n'est pas prioritaire par rapport à la vitesse de mobilité ; elle n'intervient donc pas dans la décision de basculement. Ainsi, conformément aux règles (Cf. section 5.2.1), une vitesse de mobilité de 5 m/sec entraîne un changement dans le mode de fonctionnement du protocole hybride et les nœuds du réseau adoptent ainsi le comportement de ODMRP. Le ratio de livraison des paquets de données est ainsi amélioré et les performances du protocole hybride sont pratiquement semblables à celles de ODMRP. Bien sûr, le basculement a nécessité l'utilisation de paquets de contrôle afin d'établir la nouvelle structure mais cela n'a que très peu d'effet sur les résultats.

L'utilisation d'un protocole hybride devient intéressante lorsque l'on étudie la figure 5.9 qui fait le lien entre l'énergie normalisée et la taille du groupe multicast. En effet, on s'aperçoit que la consommation d'énergie des nœuds est certes plus importante que celle de EM2NET lorsque la taille du groupe est supérieure à 20 membres mais cela est dû à un ratio de livraison très largement supérieur (Cf. figure 5.8). Il est donc important d'étudier conjointement les deux courbes afin de se rendre compte de l'efficacité du protocole hybride.

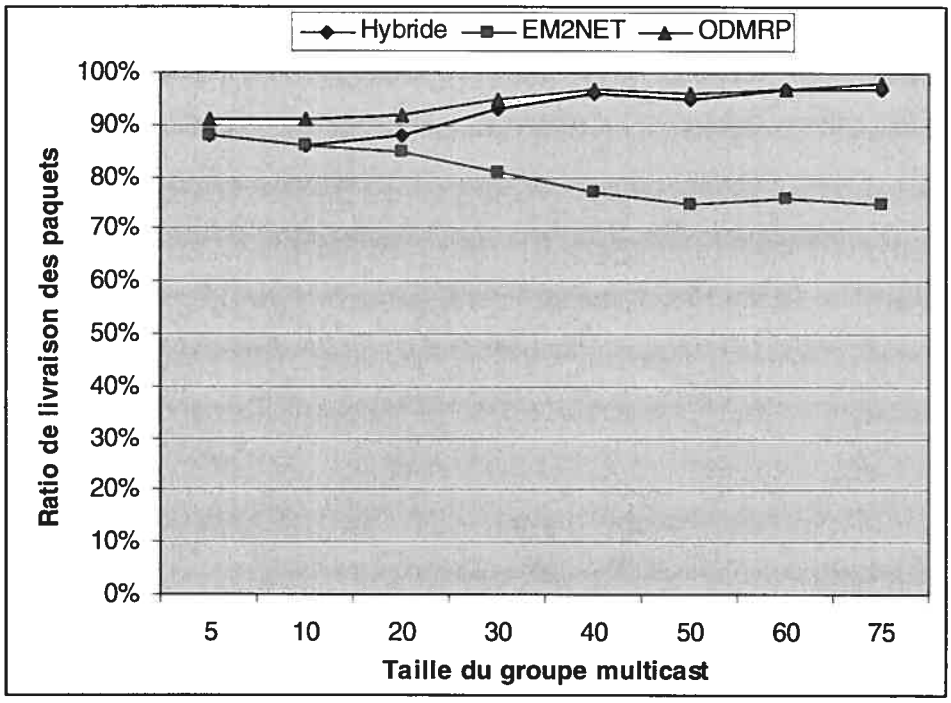

Figure 5.8. Ratio de livraison des paquets de données en fonction de la taille du groupe multicast.

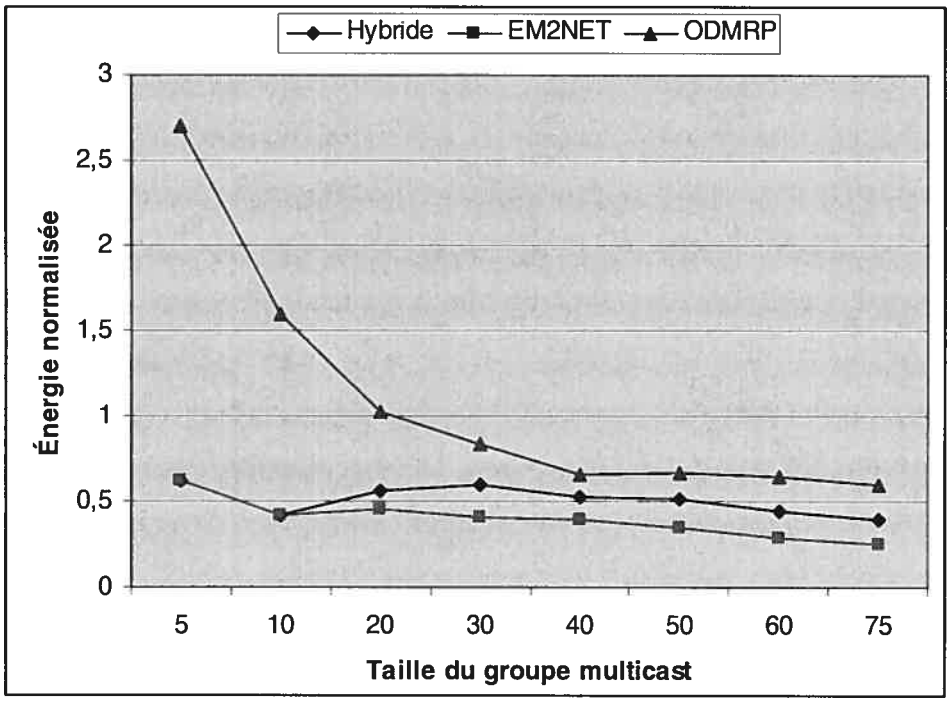

Figure 5.9. Énergie normalisée en fonction de la taille du groupe multicast.

Étudions à présent l'influence de la vitesse de mobilité sur les performances des trois protocoles. La figure 5.10 présente les résultats en terme de ratio de livraison. Notons que le groupe multicast est composé de 20 nœuds et que la charge du trafic est standardisée à 2 pkt/sec. Nous remarquons que le protocole hybride commence à s'exécuter en mode EM2NET car la vitesse de déplacement des nœuds est inférieure au seuil de 5 m/sec mais qu'un basculement intervient par la suite. Le protocole ODMRP prend ainsi le relais, ce qui engendre une amélioration du ratio de livraison des paquets de données. Toutefois, cela entraîne également une légère augmentation de l'énergie normalisée comme nous pouvons le voir sur la figure 5.11. Mais, identiquement aux courbes précédentes relatives à la taille du groupe multicast, cette augmentation s'explique par un meilleur rendement dans la livraison des données. Si le basculement n'avait pas eu lieu, l'énergie normalisée aurait en effet été moins importante car le protocole EM2NET serait celui en charge de l'exécution mais le ratio de livraison des paquets ne serait alors pas aussi élevé, d'où la nécessité d'un changement du mode de fonctionnement.

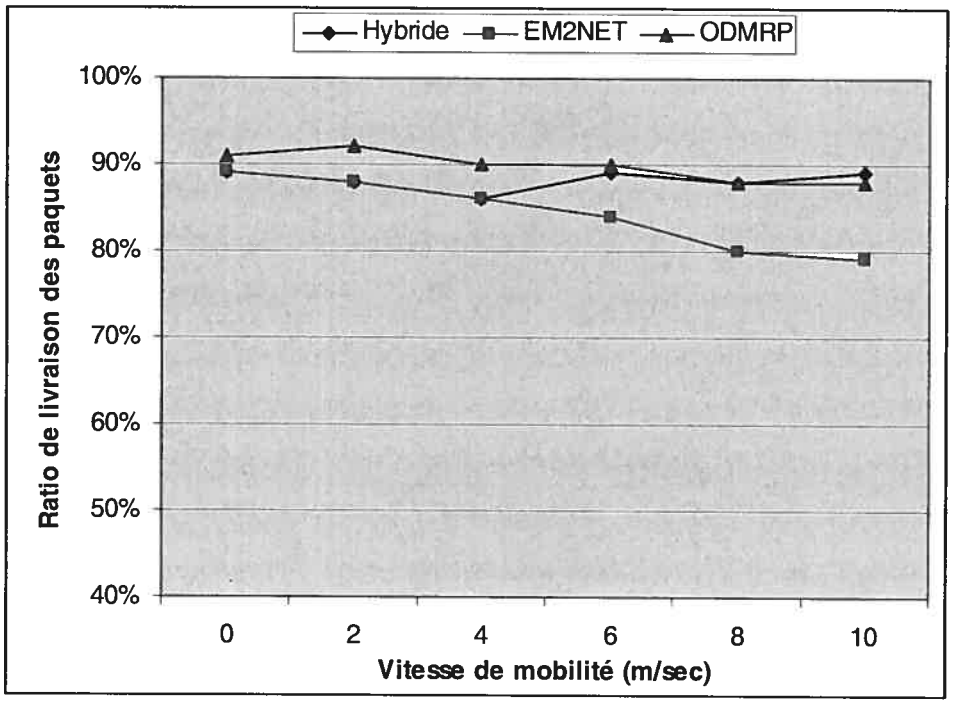

Figure 5.10. Ratio de livraison des paquets de données en fonction de la vitesse de mobilité.

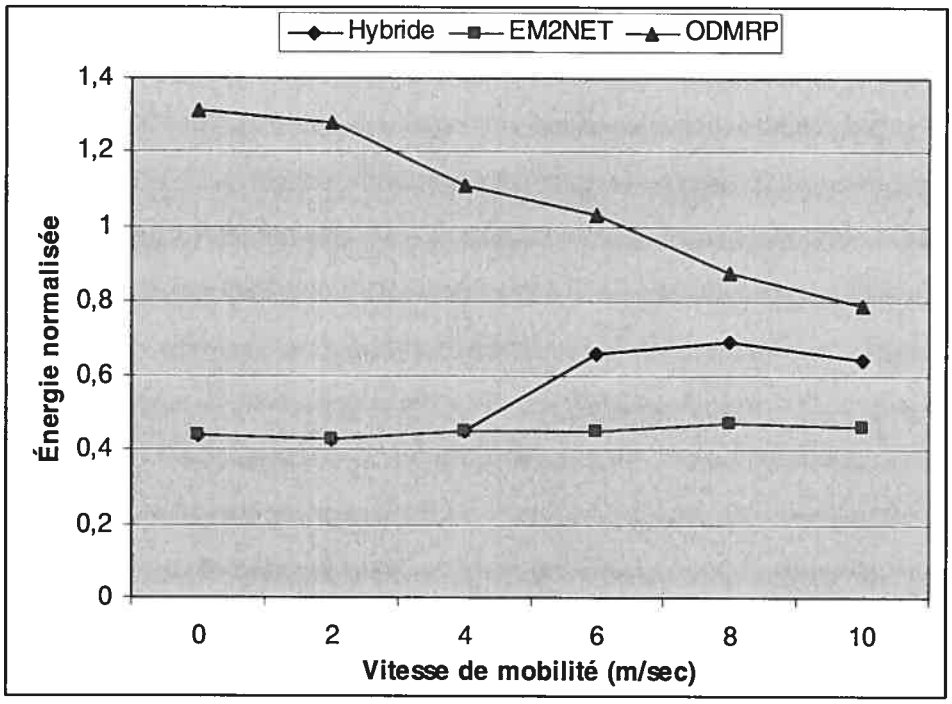

Figure 5.11. Énergie normalisée en fonction de la vitesse de mobilité.

Finalement, le ratio de livraison des paquets de données pour différentes charges de trafic est présenté sur la figure 5.12. 20 membres composent le groupe multicast et la vitesse de mobilité est établie à 5 m/sec. La charge du trafic est donc le seul critère qui intervient dans la décision de basculement. Nous commençons avec des charges de trafic de 40 pkt/sec jusqu'à atteindre 1 pkt/sec dans le but de montrer que les performances du protocole hybride sont les mêmes selon que le changement du mode de fonctionnement se fasse dans le sens EM2NET-ODMRP ou dans le sens ODMRP-EM2NET. Là encore, le basculement permet de bénéficier au moment opportun des avantages des protocoles EM2NET et ODMRP. De plus, nous nous apercevons sur la figure 5.13 que la surcharge des bytes de contrôle suit le phénomène naturel du changement de mode de fonctionnement et que la modification de la structure au cours de la simulation n'entraîne qu'une très légère augmentation du nombre de paquets de contrôle échangés. On peut donc dire que le basculement ne s'accompagne à priori pas de perte de paquets, d'où le bon ratio de livraison de données obtenu.

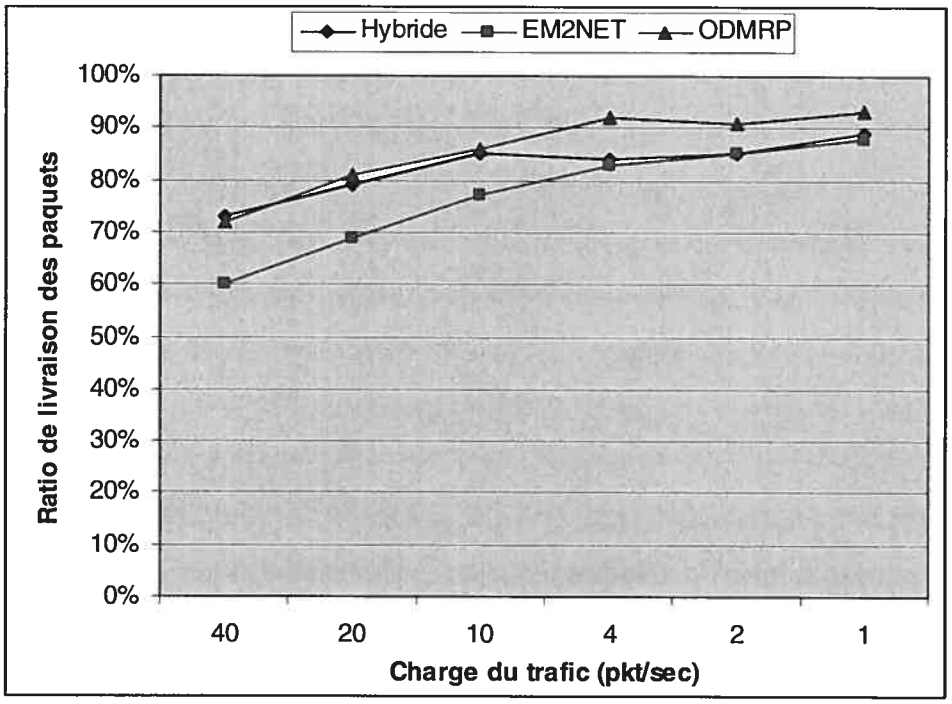

Figure 5.12. Ratio de livraison des paquets de données en fonction de la charge du trafic.

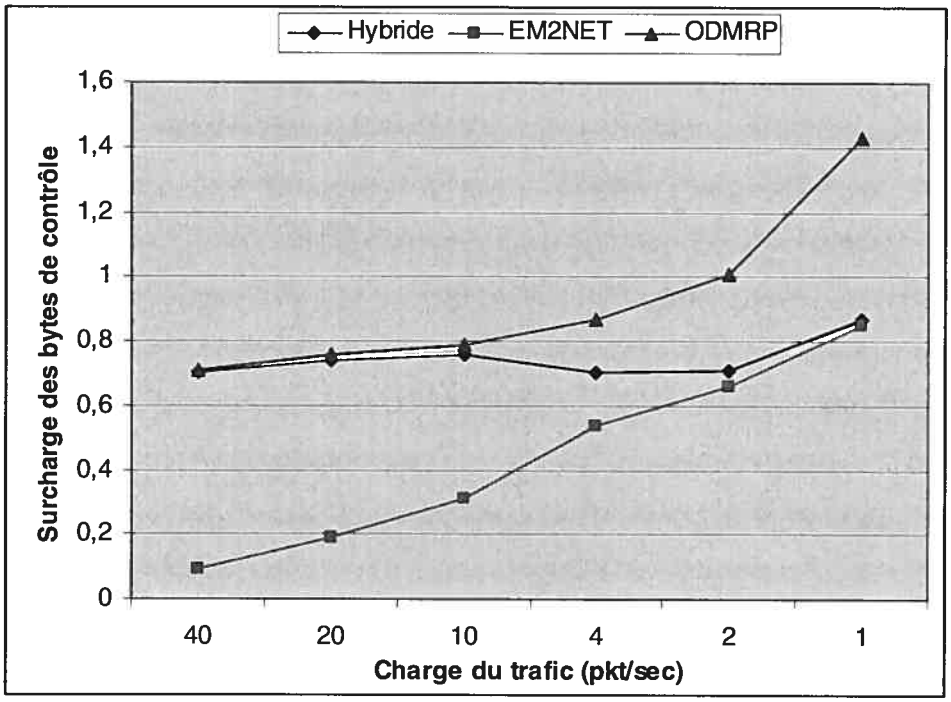

Figure 5.13. Surcharge des bytes de contrôle en fonction de la charge du trafic.

5.2.3 Conclusion

Nous venons de proposer un protocole hybride basé sur l'utilisation en alternance des protocoles EM2NET et ODMRP. La méthode employée lors du changement de mode de fonctionnement permet de nuancer les effets du basculement entre les deux modes. En effet, le ratio de livraison des paquets n'est pas affecté (ou très peu) et les paquets de contrôle nécessaires à l'établissement d'une nouvelle structure sont peu nombreux. Les simulations que nous avons menées en sont la preuve (Cf. section 5.2.2). Pour chacun des paramètres d'étude (taille du groupe multicast, vitesse de mobilité et charge du trafic), le protocole hybride assure des performances optimales quelque soit la métrique de comparaison. En particulier, il offre un bon ratio de livraison des paquets de données couplé à une énergie normalisée très faible lorsque l'on fait le parallèle avec l'utilisation des protocoles EM2NET et ODMRP exécutés séparément. Cela signifie qu'il est capable de s'adapter à l'environnement du réseau et d'adopter ainsi le comportement du protocole dont l'efficacité et la robustesse ont été démontrées dans cet environnement particulier (Cf. chapitre 4). Le protocole hybride apparaît donc comme un très bon compromis lorsque le réseau ad hoc est sujet à de fréquentes et imprévisibles modifications de ses paramètres (par exemple lorsque la session multicast peut aussi bien être de petite taille qu'à forte densité).

Chapitre 6

Conclusion

6.1 Conclusion générale

Dans ce mémoire, nous avons présenté un nouveau protocole multicast explicite pour les réseaux ad hoc mobiles, appelé EM2NET. Il est basé sur la construction d'un arbre multicast composé de nœuds particuliers, les nœuds IN, qui constituent l'épine dorsale de cet arbre. Ces nœuds permettent un balancement de la charge et cela inclue des fonctions telles que l'encapsulation, la répartition des destinations, la copie et la transmission des données. EM2NET dispose également d'une méthode pour supporter la mobilité des nœuds et les pannes. Puis nous avons mené une analyse théorique et conduit une évaluation de performances de 4 protocoles multicast pour MANETs (EM2NET, E²M, ODMRP et MAODV). Le simulateur de réseaux ns-2 nous a permis de mener des comparaisons équitables et précises des protocoles suivant un certain nombre de paramètres (vitesse de mobilité, taille du groupe multicast, charge du trafic). Les résultats ont montré que EM2NET surpassait les méthodes existantes de multicast explicite (notamment E²M) et offrait également de meilleures performances que les protocoles basés sur une structure d'arbre (MAODV). De plus, lorsque nous l'avons comparé à un protocole basé sur une structure maillée (ODMRP), nous avons prouvé qu'il permettait une conservation de l'énergie plus efficace.

Une conclusion de cette première contribution est que, dans un environnement extrêmement mobile, les protocoles basés sur une structure de

maillage offrent de meilleurs ratios de livraison de paquets car ils reposent sur la disponibilité de chemins alternatifs en cas de cassure. Cependant, cette structure redondante a un coût : un grand nombre de paquets de contrôle sont échangés, ce qui résulte en une importante consommation d'énergie. Au contraire, la méthode proposée dans EM2NET est particulièrement bien adaptée aux petits groupes multicast. Le balancement de la charge entre les nœuds IN rend le protocole plus robuste à la mobilité et la taille moyenne de l'en-tête des paquets de données est réduite. De plus, EM2NET réduit la consommation d'énergie des nœuds et préserve ainsi la durée de vie du réseau.

Dans la deuxième partie de ce mémoire, nous avons mis en avant les avantages et inconvénients du protocole EM2NET (en se basant sur les résultats préalablement établis dans la première contribution) et ainsi proposé des solutions permettant d'améliorer son efficacité. La première solution a pour but de réduire le nombre de messages de contrôle échangés au sein du réseau tandis que la seconde propose un nouveau mécanisme d'adhésion au groupe multicast. Là encore, nous avons constaté grâce aux simulations une amélioration du rendement de EM2NET lors de l'étude de métriques telles que la surcharge des bytes de contrôle, l'énergie normalisée ou encore le ratio de livraison des paquets.

Finalement, la troisième contribution a consisté en la proposition d'une méthode hybride basée sur les protocoles EM2NET et ODMRP. Ce protocole s'adapte dynamiquement à l'environnement du réseau ad hoc afin d'offrir en tout temps les meilleures performances possibles. Nous avons mené une étude de performances détaillée afin d'identifier les situations les plus favorables à l'un ou l'autre des protocoles. Ainsi, nous avons établit les règles de basculement puis nous avons proposé un mécanisme d'alternance entre les deux modes de fonctionnement. Les simulations ont démontré que celui-ci permettait d'atténuer les effets du changement de structure tout en conduisant à de meilleures performances que lors de l'utilisation séparée de chacun des protocoles EM2NET et ODMRP.

6.2 Perspectives

Comme travaux futurs, nous planifions de montrer l'influence d'un routage unicast réactif et proactif en utilisant EM2NET conjointement avec les protocoles AODV et DSR. Il serait également intéressant d'investiguer l'implémentation de EM2NET au niveau du protocole de couche MAC plutôt qu'au niveau routage comme c'est le cas actuellement [37, 38].

De plus, nous prévoyons de mener une analyse théorique plus complète en ce qui concerne les règles de basculement du protocole hybride. Par exemple, nous envisageons de déterminer analytiquement l'influence des paramètres de l'environnement (le nombre de nœuds dans le réseau, leur portée de transmission ou encore la capacité des liens) sur les valeurs de seuils utilisées lors de l'alternance entre les deux protocoles. Cela permettrait de calculer des seuils de basculement adaptés à une configuration de réseau donnée.

Bibliographie

[1] K. Savetz, N. Randall et Y. Lepage. MBone: Multicasting tomorrow's internet. *IDG Books,* 1996.

[2] M. Carron. Rapport sur le multicast. *Travail d'Étude et de Recherche de Master (TER)*, 2006.

[3] Abilene Backbone Network. http://www.ucaid.edu/abilene/home.html

[4] H. Gossain, C.M. Cordeiro et D.P. Agrawal. Multicast: wired to wireless. *IEEE Communications Magazine*, vol. 40, no. 6, pages 116-123, 2002.

[5] R. Boivie. A new multicast scheme for small groups. *IBM Research Report*, 1999.

[6] R. Boivie, N. Feldman, Y. Imai, W. Livens, D. Ooms et O. Paridaens. Explicit multicast (Xcast) basic specification. *IETF Internet Draft*, 2003.

[7] L. Ji et M.S. Corson. Differential Destination Multicast – A MANET multicast routing protocol for small groups. *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'01)*, pages 1192-1202, 2001.

[8] H. Gossain, C.M. Cordeiro, K. Anand et D.P. Agrawal. E²M: A scalable explicit multicast protocol for MANETs. *Proceedings of the IEEE International Conference on Communications (ICC'04)*, pages 3628-3632, 2004.

[9] S.-J. Lee, W. Su, M. Gerla et R. Bagrodia. A performance comparison study of ad hoc wireless multicast protocols. *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'00)*, 2000.

[10] C.M. Cordeiro, H. Gossain et D.P. Agrawal. Multicast over wireless mobile ad hoc networks: present and future directions. *IEEE Network Magazine, Special Issue on Multicasting: An Enabling Technology*, vol. 17, no. 1, pages 52-59, 2003.

[11] A. Benslimane. EM2NET: an Explicit Multicast for MANET. *Proceedings of the ACM International Wireless Communications and Mobile Computing Conference (IWCMC'06)*, pages 431-436, 2006.

[12] Network Simulator (Version 2). http://www.isi.edu/nsnam/ns

[13] S.-J. Lee, W. Su et M. Gerla. On-demand multicast routing protocol in multihop wireless mobile networks. *ACM/Kluwer Mobile Networks and Applications*, pages 441-453, 2002.

[14] E.M. Royer et C.E. Perkins. Multicast operation of the ad hoc on-demand distance vector routing protocol. *Proceedings of the ACM/IEEE MOBICOM'99*, pages 207-218, 1999.

[15] H. Moustafa. Routage unicast et multicast dans les réseaux mobiles ad hoc. *Thèse Informatique et Réseaux ENST/NFRES*, 2004.

[16] E.M. Royer et C.E. Perkins. Ad hoc on-demand distance vector (AODV) routing. *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99),* 1999.

[17] C.W. Wu et Y.C. Toy. AMRIS: a multicast protocol for ad hoc wireless networks. *Proceedings of the IEEE MILCOM '99*, 1999.

[18] J. Xie, M. Liu, A. McAuley et R. Talpade. AMRoute: Ad hoc Multicast Routing protocol. *Mobile Networks and Applications (MONET),* 2002.

[19] K. Viswanath, K. Obraczka et G. Tsudik. Exploring mesh and tree-based multicast routing protocols for MANETs. *IEEE Transactions on Mobile Computing*, vol. 05, no. 1, pages 28-42, 2006.

[20] S.-J. Lee, W. Su et M. Gerla. Wireless ad hoc multicast with mobility prediction. *Proceedings of the IEEE International Conference on Computer Communications and Networks (ICCCN'99)*, pages 4-9, 1999.

[21] J.J. Garcia-Luna-Aceves et E.L. Madruga. The core-assisted mesh protocol. *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pages 784-792, 1999.

[22] E.L. Madruga et J.J. Garcia-Luna-Aceves. Scalable multicasting: the core-assisted mesh protocol. *ACM Baltzer Mobile Networks and Applications,* 1999.

[23] M.K. Shin, Y.J. Kim, K.S. Park et S.G. Kim. Explicit multicast extension (Xcast+) for efficient multicast packet delivery. *ETRI Journal*, vol. 23, no. 4, pages 202-204, 2001.

[24] A. Boudani et B. Cousin. SEM: A new small group multicast routing protocol. *Proceedings of the IEEE International Conference on Telecommunications (ICT'03),* 2003.

[25] T. Kunz et E. Cheng. Multicasting in ad-hoc networks: comparing MAODV and ODMRP. *Proceedings of the Workshop on Ad hoc Communications,* 2001.

[26] CMU Monarch Project. Mobility Extensions to ns-2. http://www.monarch.cs.cmu.edu

[27] H. Wu, S. Cheng, Y. Peng, K. Long et J. Ma. IEEE802.11 distributed coordination function (DCF): analysis and enhancement. *Proceedings of the IEEE International Conference on Communications (ICC'02),* 2002.

[28] IEEE Computer Society LAN MAN Standards Committee. Wireless LAN Medium Access Protocol (MAC) and Physical Layer (PHY) Specification. *IEEE Standard 802.11*, 1997.

[29] D.B. Johnson et D.A. Maltz. Dynamic source routing in ad hoc wireless networks. *Mobile Computing*, pages 153-181, 1996.

[30] T. Clausen, P. Jacquet et L. Viennot. Comparative study of routing protocols for mobile ad-hoc networks. *Proceedings of IFIP Med-Hoc-Net 2002,* 2002.

[31] J. Broch, D.A. Maltz, D.B. Johnson, Y.-C. Hu et J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. *Proceedings of the IEEE/ACM MOBICOM'98*, pages 85-87, 1998.

[32] S.R. Das, C.E. Perkins, E.M. Royer et M.K. Marina. Performance comparison of two on-demand routing protocols for ad hoc networks. *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'00),* 2000.

[33] S.R. Das, R. Castañeda, J. Yan et R. Sengupta. Comparative performance evaluation of routing protocols for mobile, ad hoc networks. *Proceedings of the IEEE International Conference on Computer Communications and Networks (IC3N'98),* pages 153-161, 1998.

[34] Internet Engineering Task Force (IETF) Mobile Ad Hoc Networks (MANET) Working Group Charter. http://www.ietf.org/html.charters/manet-charter.html

[35] T. Kunz. Multicasting in mobile ad-hoc networks: achieving high packet delivery ratios. *Proceedings of the 2003 Center for Advanced Studies Conference (CAS)*, pages 156-170, 2003.

[36] J. Biswas, M. Barai et S.K. Nandy. Efficient hybrid multicast routing protocol for ad-hoc wireless networks. *Proceedings of the IEEE Conference on Local Computer Networks (LCN'04)*, 2004.

[37] M. Takai, J. Martin et R. Bagrodia. Effects of wireless physical layer modeling in mobile ad hoc networks. *Proceedings of the ACM MobiHoc'01*, 2001.

[38] H. Gossain, N. Nandiraju, K. Anand et D.P. Agrawal. Supporting MAC layer multicast in IEEE 802.11 based MANETs: issues and solutions. *Proceedings of the IEEE Conference on Local Computer Networks (LCN'04)*, 2004.