Titre: Title:	Configuration des noeuds d'un réseau mobile ad hoc
Auteur: Author:	Abdellatif Ezzouhairi
Date:	2004
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Ezzouhairi, A. (2004). Configuration des noeuds d'un réseau mobile ad hoc [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7481/

UNIVERSITÉ DE MONTRÉAL

CONFIGURATION DES NŒUDS D'UN RÉSEAU MOBILE AD HOC

ABDELLATIF EZZOUHAIRI
DÉPARTEMENT DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
Août 2004

Your file Votre référence
ISBN: 0-612-97946-6
Our file Notre référence
ISBN: 0-612-97946-6

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

CONFIGURATION DES NŒUDS
D'UN RÉSEAU MOBILE AD HOC

Présenté par : EZZOUHAIRI Abdellatif
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen composé de :

Mme BOUCHENEB Hanifa, Doctorat, présidente

M. QUINTERO Alejandro, Doct., directeur de recherche

Mme NICOLESCU Gabriela, Doctorat, membre

Dédicace

À ma très chère femme,

à ma fille,

à toute ma famille.

Abdellatif EZZOUHAIRI

Remerciements

Je tiens à exprimer mes sincères remerciements à mon directeur de recherche, le professeur **Alejandro Quintero**, pour la qualité de son encadrement, ses suggestions et sa disponibilité.

Je remercie également le professeur **Samuel Pierre** Directeur du LARIM, pour son aide et son accueil.

Mes remerciements vont aussi à tous les membres du LARIM, pour leur soutien et pour l'ambiance de travail.

Enfin, je tiens à exprimer ma gratitude à tous ceux qui ont participé de près ou de loin à la réalisation de ce travail, et en particulier à M. J. Abrache.

Résumé

L'architecture des réseaux mobiles conventionnels est traditionnellement bâtie autour d'une infrastructure fixe qui constitue le cœur du réseau, et d'une partie mobile constituée des périphériques des usagers. En revanche, dans le cas des réseaux mobiles ad hoc, aucune infrastructure fixe n'est nécessaire, et les unités mobiles qui forment ce genre de réseau communiquent uniquement à travers leurs interfaces radio. Suite à ce nouveau contexte de mobilité, une multitude de problèmes apparaissent et de nouvelles solutions s'avèrent par conséquent nécessaires. Parmi ces problèmes on trouve celui de la configuration, principale préoccupation de ce mémoire, qui consiste à proposer un mécanisme d'allocation d'adresses IP tout en tenant compte du comportement à la fois dynamique et aléatoire des noeuds mobiles.

Jusqu'à présent, quelques approches de configuration pour les réseaux ad hoc ont été proposées. Ces approches peuvent être classées en deux catégories : méthodes avec détection de conflits et méthodes sans détection de conflits. L'idée des approches de configuration avec détection de conflits consiste à proposer une adresse IP et ensuite à demander l'avis des autres composants du réseau pour s'assurer de son unicité. Par contre, dans l'autre catégorie de méthodes, la configuration se fait de façon directe sans que l'avis du reste du réseau ne soit sollicité. Cependant, ces approches de configuration présentent de sérieuses faiblesses concernant aussi bien l'unicité de l'adresse proposée qui reste très souvent conditionnée, que leurs aptitudes à traiter les principaux problèmes liés à la nature dynamique de ce type de réseau, tels que la gestion des départs, la récupération des adresses, le partitionnement et la fusion.

Notre approche de configuration appelée APM (Autoconfiguration Protocol for MANETs), permet à une nouvelle unité mobile d'intégrer un réseau ad hoc déjà en fonction, en demandant une adresse IP à l'un de ses voisins immédiats. Quant à la gestion des départs, la récupération des adresses non utilisées, le partitionnement et la fusion, ils sont traités de façon centralisée par un nœud particulier appelé nœud de configuration.

Afin de mettre en exergue notre approche de configuration, nous avons réalisé des simulations avec GLOMOSIM (Global Mobile Simulator). Tout au long de nos expériences, nous avons mesuré d'une part le temps pour satisfaire une demande de configuration (latence) et d'autre part le trafic généré, en terme de messages, par cette demande. En guise de comparaison, nous avons aussi implémenté une des principales méthodes de configuration basées sur la détection de conflits, en l'occurrence, la méthode de [Nesargi 2002]. Les résultats montrent que le protocole APM affiche de bonnes performances comparativement à celui de Nesargi. Et ce même en présence de départs massifs des unités mobiles dans le réseau mère.

Abstract

The architecture of the conventional mobile networks is traditionally built around a fixed infrastructure which constitutes the core network, and a mobile part made up of users peripherals. However, in the case of the ad hoc mobile networks, no fixed infrastructure is necessary, and the mobile units, which form this kind of network, communicate only through their radio interfaces. Following this new mobility context, a multitude of problems appear, and consequently, new solutions prove to be necessary. Among these problems, we can enumerate: routing, neighbors detection, services detection, energy consumption, flooding and configuration, which is the main concern of this memory. Configuration problems consist of proposing a mechanism to assign IP addresses to new mobile units that need to integrate with an ad hoc network, and deal with the dynamic and random behavior of those units.

Until now, some ad hoc configuration approaches have been proposed, which can be classified in two categories: methods with conflict detection and methods without conflict detection. The idea of the approaches of configurations with conflict detection consists of proposing an IP address, and then, verifying its uniqueness by asking a permission of the other components of the network. Whereas, in the other category of methods, the configuration is made in a direct way (the opinion of the network is not requested). However, all of these approaches have weaknesses such as addresses recuperation and effective resolution of partitioning and merging problems.

Our configuration approach called APM (Autoconfiguration Protocol for MANETs) allows a new mobile unit to integrate with an ad hoc network, by requiring an IP address from its immediate neighbors. The management of the problems related to the dynamic behavior of the network, such as nodes departure, partitioning and merge is realized by a special node called configuration node.

In order to put forward our configuration approach, we carried out simulations with GLOMOSIM (Global Mobile Simulator). Throughout our experiments, we measured the time to satisfy a configuration request (latency) and the traffic in term of the number of messages generated by this request. We have also implemented the configuration mechanism of Nesargi [Nesargi 2002] for comparison. The results show that our protocol APM posts better performance comparatively to the Nesargi protocol, and this, despite massive departures of mobile units in the network.

Table des matières

Liste des tableaux

Liste des figures

Liste des sigles et abréviations

APM	:	Autoconfiguration Protocol for MANETs
BS	:	Base Station
BSC	:	Base Station Controller
BTS	:	Base Transceiver Station
CBR	:	Constant Bit Rate
DCDP	:	Dynamic Configuration and Distribution Protocol
DHCP	:	Dynamic Host Configuration Protocol
DRCP	:	Dynamic Registration and Configuration Protocol
EUI	:	Extended Universal Identifier
GLOMOSIM	:	Global Mobile information system Simulator
HLR	:	Home Location Register
IANA	:	Internet Assigned Numbers Authority
IEEE	:	Institute of Electrical and Electronics Engineers
IP	:	Internet Protocol
MAC	:	Medium Access Control
MANET	:	Mobile Ad hoc Network
MSC	:	Mobile Swiching Center
NC	:	Nœud de Configuration
PARSEC	:	Parallel Simulation Environment for Complex systems
PDA	:	Personal Data Assistant
PPP	:	Point-to-Point Protocol
RTC	:	Réseau Téléphonique Commuté
UM	:	Unité Mobile
VLR	:	Visitor Location Register
WLAN	:	Wireless Local Area Network

Chapitre 1
Introduction

L'informatique mobile a connu d'importants progrès ces dernières années, notamment avec l'apparition sur le marché des réseaux mobiles de troisième génération. Les progrès enregistrés sont axés principalement autour du débit et les services offerts aux usagers. Le débit est devenu de plus en plus important même en présence d'une grande mobilité. Et les services offerts, ont tendance à concurrencer d'avantage ceux offerts par les réseaux filaires (multimédia, internet, etc.). Quant à l'architecture, elle demeure encore basée sur la dualité : partie mobile et partie fixe. Malgré toutes ces avancées, il existe des situations où l'installation d'un réseau mobile conventionnel n'est pas possible pour des raisons de commodité (cas de sinistres par exemple), de coût et de performance. Les réseaux mobiles ad hoc constituent une alternative aux réseaux mobiles traditionnels dans de telles situations. En effet, un réseau mobile ad hoc ne se base sur aucune infrastructure fixe et les unités mobiles qui le constituent communiquent uniquement à travers leurs interfaces radio. Ce nouveau cadre de mobilité donne lieu à une multitude de problèmes. A titre d'exemple, on cite le problème du routage qui nécessite dans ce cas de nouveaux mécanismes pour pouvoir acheminer des données d'une unité mobile à une autre. On cite également le problème de configuration, principale préoccupation de ce mémoire, qui traite de l'attribution des adresses IP en tenant compte de ces nouvelles contraintes de mobilité.

Dans ce qui suit, nous allons définir quelques concepts de base, puis nous présenterons les grandes lignes de notre problématique, ensuite nous exposerons nos objectifs de recherche et nous finirons par une description du plan de ce mémoire.

1.1 Définitions et concepts de base

→ Mobilité : désigne la capacité d'accéder, à partir de n'importe quel endroit, à l'ensemble des services normalement disponibles dans un environnement fixe [Samuel 2003].

→ Informatique mobile : réfère à la possibilité pour des usagers munis de périphériques portables ou d'ordinateurs mobiles d'accéder à des services et des applications évoluées, à travers une infrastructure partagées de réseau, indépendamment de leurs localisation physique ou de leur comportement de mouvement [Samuel 2003].

→ Unité mobile : désigne tout composant susceptible de se déplacer sans contraintes, et de communiquer uniquement à travers son interface radio. On peut trouver d'autres synonymes tel que : nœud mobile, composant mobile, terminal mobile, périphérique mobile, etc.

→ Réseau mobile ad hoc : on désigne par réseau mobile ad hoc ou MANET (Mobile Ad hoc Network), un ensemble de terminaux mobiles formant un réseau et opérant sans la présence d'une infrastructure fixe [Samuel 2003].

→ Configuration : la configuration désigne, d'une façon générale, le fait d'attribuer des paramètres à un composant pour le rendre adapté à accomplir certaines tâches.

→ Autoconfiguration : réfère à toute action visant à accomplir une opération de configuration par soi-même.

1.2 Exposé de la problématique

Comme nous l'avons précédemment mentionné, les réseaux mobiles ad hoc se proposent comme une alternative aux réseaux mobiles conventionnels dans certaines situations. Ceci est le cas par exemple lors d'une catastrophe naturelle où il serait impossible d'attendre jusqu'au rétablissement du service mobile courant pour pouvoir organiser des opérations de secours. Et, d'une manière générale, ce genre de réseau peut être déployé à chaque fois qu'on a besoin d'une plate forme mobile à moindre coût ou

lorsque la performance ne constitue pas un critère de première importance. Les applications qui peuvent s'intéresser aux réseaux mobiles ad hoc, relèvent de différents domaines. À titre d'exemple, on peut citer celui de l'enseignement et de la recherche scientifique où des laboratoires mobiles temporaires peuvent être utilisés dans des zones difficiles d'accès. Avec la monté grandissante des équipements mobiles à domicile, les applications en domotique constituent également un grand champ d'application pour ce genre de réseau. Cependant, ce nouveau cadre de mobilité soulève de nombreux problèmes qui empêchent encore le concept des réseaux ad hoc d'être concurrentiel. Parmi ces problèmes, on trouve celui de la gestion de l'attribution des adresses IP pour les unités mobiles. Ce problème est souvent connu dans la littérature sous le nom de configuration ou autoconfiguration. En effet, pour que des composantes mobiles, faisant partie d'un MANET, puissent communiquer et échanger des informations, elles doivent posséder au préalable des adresses IP. Dans le cas des réseaux filaires, la configuration est réalisée habituellement par l'une des méthodes suivantes :

- Manuellement, surtout lorsqu'il s'agit d'un petit réseau;
- moyennant des protocoles de configuration tel que DHCP [Droms 1997] pour des réseaux de taille importante;
- avec parfois des protocoles tel que PPP [Simpson 1994] ou mobile IP [Perkins 2000],
- à l'aide d'un mécanisme d'autoconfiguration comme c'est le cas d'IPv6 [Cizault 2002].

Mais, dans le cas d'un réseau mobile ad hoc, on ne peut pas appliquer les approches de configuration citées plus haut, étant donné qu'elles se basent toutes sur une infrastructure fixe.

Par conséquent, de nouvelles approches de configuration s'avèrent nécessaires afin de permettre aux nœuds d'un MANET de communiquer entre eux. Une telle approche doit être en mesure d'attribuer des adresses IP aux unités mobiles désirant intégrées un réseau déjà en fonction, tout en tenant compte de leurs comportements à la fois dynamiques et aléatoires. Malheureusement, la configuration de ce genre de réseau ne se limite pas

uniquement à l'allocation d'adresses IP. En effet, d'autres défis apparaissent en raison de l'absence d'une infrastructure fixe. Parmi ceux-ci, on peut citer la gestion des départs des nœuds et la récupération de leurs adresses en vue d'une prochaine réutilisation. Il faut par ailleurs faire face au problème de partitionnement qui consiste en la division d'un réseau ad hoc en une ou plusieurs parties. Ce phénomène peut arriver de façon accidentelle ou suite à des départs massifs de noeuds. La fusion constitue également un sérieux défi à soulever, car à cause du caractère mobile des réseaux ad hoc, on se retrouve souvent avec des situations où deux ou plusieurs réseaux aient une intersection non vide entre eux et on doit à ce moment proposer un moyen pour que les nœuds puissent communiquer correctement au sein d'une même zone géographique. La résolution de tous ses problèmes et bien d'autres va constituer la principale préoccupation de ce mémoire.

Les recherches qui ont été menées jusqu'à présent sur la problématique de configuration des réseaux mobiles ad hoc peuvent être classées en deux catégories : des méthodes de configuration avec détection de conflits et des méthodes de configuration sans détection de conflits. Les méthodes de configuration avec détection de conflits, proposent à une nouvelle entité mobile, une adresse IP, ensuite elles en vérifient l'unicité en demandant l'accord de tous les autres composants du réseau. Si un nœud pose son veto à cette adresse, le processus d'allocation d'adresse est relancé. On trouve ce genre d'approche, chez [Perkins 2001], [Nesagi 2002] et [Weniger 2002]. Ce genre de méthodes, présente l'avantage d'être peu complexe et peut être implanté facilement. Cependant, le fait qu'il se base sur la diffusion, occasionne d'importants délais d'attente chez les nœuds demandeurs d'adresses IP, ainsi qu'une augmentation considérable de la signalisation dans le réseau. Quant aux méthodes sans détection de conflits, elles proposent des mécanismes d'attribution d'adresses IP de façon directe, c'est à dire qu'on n'a plus besoin de l'avis du reste du réseau pour garantir l'unicité de l'adresse proposée. On évite dans ce cas le recours à la diffusion, ce qui se traduit par de meilleurs délais d'attente pour les requêtes de configuration d'une part et par une signalisation réduite d'autre part. De telles approches de configuration ont été proposées par [Mohsin 2002] et

[Zhou 2003]. Cependant, elles présentent la limitation de ne pas prévoir de mécanisme pour la gestion de la récupération des adresses et parfois l'unicité de celles-ci est conditionnée comme c'est le cas avec la méthode de [Zhou 2003].

Chacune des approches proposées apportent des solutions à certains problèmes alors que d'autres restent non résolus, ce qui fait des réseaux ad hoc un domaine de grand intérêt pour les chercheurs.

1.3 Objectifs de la recherche

Notre objectif durant ce travail est de proposer un protocole d'autoconfiguration pour les réseaux mobiles ad hoc. Ce protocole doit garantir l'unicité de l'adresse, autrement dit, il ne doit pas y avoir deux unités mobiles avec une même adresse IP. Il doit être accessible à tous les nouveaux nœuds qui veulent intégrer le réseau. Ceci signifie que, le service de configuration doit être disponible en permanence sur le réseau. Il doit aussi proposer un mécanisme de gestion des départs et de récupération d'adresses, ainsi que le traitement des problèmes de partitionnement et de fusion. Plus précisément notre objectif est de :

- Préciser les spécifications que doit respecter notre protocole à la lumière de ce qui a été proposé jusqu'à présent et selon ce qu'elles doivent être dans le futur.
- Proposer un protocole d'autoconfiguration pour les réseaux mobiles ad hoc répondant aux précédentes exigences.
- Garder les avantages des autres méthodes de la littérature, et éviter leurs limitations au maximum.
- Réaliser des simulations pour analyser les performances de notre protocole.
- Comparer notre mécanisme de configuration à l'une des méthodes basées sur la détection de conflits.
- Discuter et analyser les résultats obtenus.

1.4 Plan du mémoire

Ce mémoire est structuré, en plus du présent chapitre, de quatre autres chapitres. Dans le chapitre 2 nous présentons un survol des principales méthodes de configuration

relatives aux réseaux mobiles ad hoc. Le chapitre 3 est consacré à la description détaillée de notre protocole d'autoconfiguration, et discute entre autres les principaux apports dudit protocole par rapport aux récentes études. Le chapitre 4 traite de tous les éléments nécessaires à l'implémentation ainsi que des descriptions de toutes les expériences de simulations et des commentaires sur les différents résultats. Enfin, dans le chapitre 5, nous concluons par une discussion sur le protocole proposé, en précisant son apport, ses limites, ainsi que les éventuelles futures pistes de recherche.

Chapitre 2
Méthodes d'autoconfiguration pour MANETs

Dès que les chercheurs ont commencé à s'intéresser au concept des réseaux mobiles ad hoc, de nombreux problèmes ont surgi, et les efforts se sont multipliés pour faire sortir ce nouveau concept du cadre du laboratoire à l'exploitation réelle. De ces efforts témoignent, par exemple, la multitude d'études menées sur les différents problèmes liés aux réseaux mobiles ad hoc. Parmi ces problèmes, on trouve le routage [Abolhasan 2003], le "flooding" [Zhu 2003], la consommation d'énergie [Kadayif 2004], la détection de voisins [Krco 2003] et de services [Ulas 2003], etc. Ces problèmes ne sont hélas pas les seuls. Comme nous l'avons mentionné dans le chapitre 1, le problème de configuration reste aussi un important défit à soulever. Ce chapitre sera entièrement consacré à la présentation des différentes méthodes de configuration relatives aux réseaux mobiles ad hoc. Plus exactement, nous allons, dans un premier temps, présenter de façon générale les environnements mobiles. Ensuite, nous introduirons les réseaux mobiles ad hoc et leurs caractéristiques. Enfin, nous donnerons une synthèse des approches d'autoconfiguration qui traite du problème d'allocation d'adresses dans les réseaux mobiles ad hoc.

2.1 Les environnements mobiles

Un environnement mobile fait référence à tout système où des usagers munis d'unités mobiles peuvent accéder à des services indépendamment de leur position géographique ou de leur mouvement. Dans la pratique, un environnement mobile peut être formé d'une infrastructure fixe (réseau filaire) et de périphériques mobiles qui interagissent avec cette infrastructure. Ceci peut être le cas d'un réseau cellulaire ou d'un réseau local sans fil WLAN [IEEE 2003]. Il se peut aussi que l'environnement

mobile soit dépourvu d'infrastructure fixe comme c'est le cas des réseaux mobile ad hoc [Samuel P. 2003]. Dans ce qui suit, nous allons voir avec plus de détails ces principales catégories d'environnements mobiles.

2.1.1 Environnements mobiles avec infrastructure fixe

Lorsqu'on parle de réseau mobile ou d'environnement mobile, on fait souvent référence au réseau cellulaire. Ce genre de réseau représente, en effet, un exemple concret d'environnement mobile avec infrastructure fixe. Afin de comprendre d'avantage ce type d'environnements mobiles, nous allons essayer de démystifier l'architecture d'un réseau cellulaire de deuxième génération (2G). Dans ce genre de réseau, le territoire couvert (ou zone de couverture), est découpé en petites surfaces géographiquement limitées qu'on appelle cellule [Samuel 2003]. Chaque cellule est associée à une station de base ou BTS (Base Transceiver Station). Celle ci, intègre une antenne qui assure la transmission radio et la signalisation à l'intérieur de la cellule. Les stations de base sont à leur tour reliées à des contrôleurs de station de base ou BSC (Base Station Controller) qui gèrent les ressources radio ou les bandes passantes des stations de base associées. Les différentes stations de base sont interconnectées entre elles et reliées avec d'autres éléments fixes du réseau par des commutateurs MSC (Mobile Switching Center) comme le montre la figure 2.1.

2.1.2 Environnements mobiles sans infrastructure

Les récentes évolutions dans le domaine de la communication sans fil, ont fait apparaître sur le marché de nouvelles composantes mobiles, en l'occurrence des téléphones portables de troisième génération, des PDA (Personal Data Assistant), des laptops, etc. L'abondance de ces équipements laisse penser à toutes sortes de connexions et d'applications. Cette situation a donné lieu à des environnements mobiles entièrement dynamiques. Ce genre d'environnement ne possède aucune infrastructure fixe, les unités mobiles qui le constituent se déplacent tout en communiquant entre elles. La communication se fait uniquement à travers leurs interfaces radio [GSM Radio interface 1996].

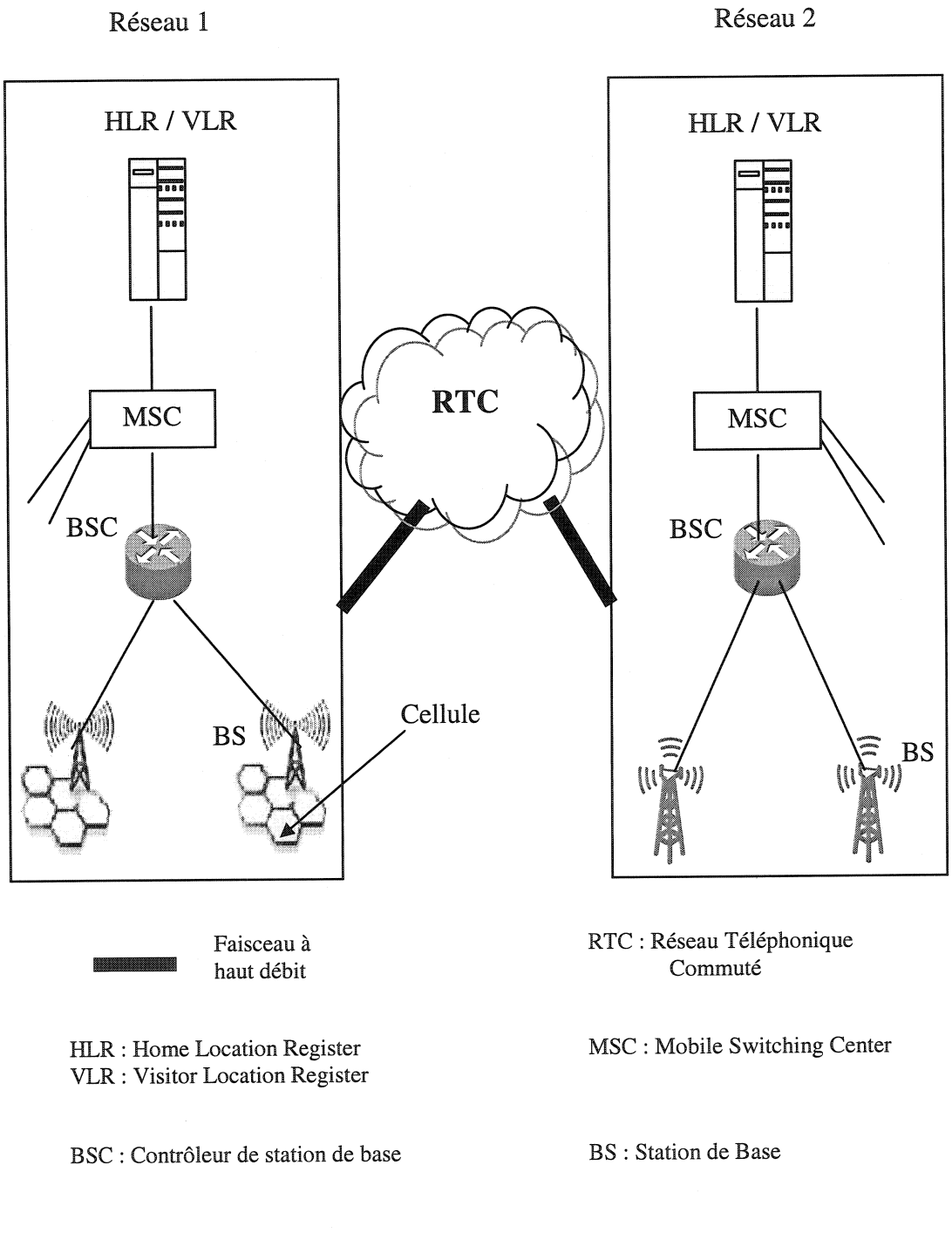


Figure 2.1 : Exemple d'environnement mobile avec infrastructure fixe

La figure 2.2 montre un exemple d'environnement mobile sans infrastructure.

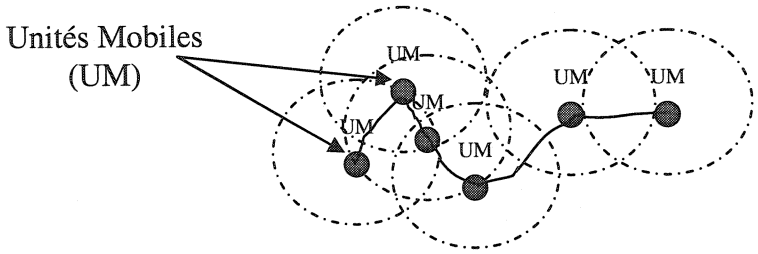


Figure 2.2 : Exemple d'environnement mobile sans infrastructure

L'absence d'infrastructure (réseau filaire ou stations de base), oblige les unités mobiles à se comporter comme des routeurs. Ils vont donc participer à la découverte et à la maintenance des chemins pour les autres hôtes du réseau. Les réseaux mobiles ad hoc représentent l'exemple type de réseaux mobiles sans infrastructure fixe.

- **Définition d'un réseau mobile ad hoc**

Un réseau mobile ad hoc, appelé également MANET (Mobile Ad Hoc NETwork), est constitué d'une population, relativement dense, d'unités mobiles qui peuvent se déplacer de façon aléatoire, et dont le seul moyen de communication est l'utilisation des interfaces sans fil.

2.1.3 Applications des réseaux mobiles ad hoc

Les réseaux mobiles ad hoc ont l'avantage d'être peu coûteux, se déploient facilement et ne nécessite aucune infrastructure de base. De ce fait, les applications de ce nouveau concept de réseau deviennent de plus en plus importantes et diversifiées. On peut citer à titre d'exemple le cas d'un sinistre où le déploiement d'un MANET permettrait l'organisation et la coordination des opérations de secours. On cite aussi l'exemple de laboratoires de recherche mobiles, qui peuvent se déployer dans les régions éloignées. On trouve aussi des applications dans le domaine de transport dont le but est de promouvoir des réseaux de véhicules intelligents. Un autre créneau d'applications des réseaux ad hoc se manifeste dans une utilisation conjointe avec les réseaux de sensors. Ceci est le cas par exemple de certaines applications militaires où des composantes

mobiles équipées de capteurs peuvent être utilisées pour collecter des informations sur les mouvements d'une armée adverse, et détecter éventuellement la présence de substances nocives. On trouve aussi dans ce sens des applications de surveillance sans fils, de détection environnementale, etc.

2.1.4 Caractéristiques des réseaux ad hoc

Vu leur comportement dynamique, les réseaux mobiles ad hoc possèdent certaines caractéristiques qui rendent cette catégorie de réseaux un peu différente de ce qu'on a l'habitude de voir. Parmi ses caractéristiques on trouve :

- **Topologie dynamique** : Les unités mobiles du réseau, se déplacent d'une façon libre et arbitraire. Par conséquent, la topologie du réseau peut changer, à des instants imprévisibles, d'une manière rapide et aléatoire.
- **Bande passante limitée** : La communication des réseaux mobiles ad hoc est basée sur l'utilisation des interfaces sans fils, ceci rend les débits de transmission différents d'un lien à l'autre. Comparativement aux réseaux filaires, les débits des MANETs demeurent modestes.
- **Contraintes d'énergie** : Les hôtes mobiles sont alimentés par des sources d'énergies autonomes comme les batteries ou les autres sources consommables. De ce fait les unités mobiles ont une autonomie énergétique très restreinte. Le paramètre d'énergie est donc très influant dans ce genre de réseau.
- **Sécurité physique limitée** : Les réseaux ad hoc, comparativement aux réseaux filaires, sont plus exposés aux attaques car par définition il n'y a aucune infrastructure physique permanente qui peut assurer des contrôles de sécurité.
- **Absence d'infrastructure** : Par définition les réseaux ad hoc ne reposent sur aucun support physique préalablement établi.

2.1.5 Pourquoi l'autoconfiguration

Les réseaux informatiques ont récemment connu une évolution spectaculaire aussi bien sur le plan des infrastructures que sur celui des services. Cette révolution a été suivie par une apparition massive de toute sorte de périphériques réseaux, ce qui a augmenté considérablement le nombre d'équipements connectés ou susceptibles de l'être.

Pour qu'un composant puisse communiquer au sein d'un réseau, il faut qu'il dispose d'une adresse IP. Celle-ci est lui attribuée, traditionnellement, de façon manuelle surtout avec des réseaux de petite taille. Cependant cette manière de faire n'est plus opérationnelle avec des réseaux de grande taille. L'autoconfiguration se propose alors comme une alternative pour remédier à cet handicap. On entend donc par autoconfiguration, le fait de doter le réseau de mécanismes de façon à ce qu'il assure par lui-même la tâche de configuration.

2.2 Méthodes d'autoconfiguration pour les MANETs

Comme nous l'avons déjà souligné, le problème de configuration dans les réseaux mobiles ad hoc consiste à attribuer des adresses IP aux unités mobiles désirant intégrer un MANET déjà en fonction et à gérer les problèmes relatifs au caractère mobile de ce genre de réseau. De nombreuses études ont été menées jusqu'à présent pour apporter des éléments de solution à ce problème. La plupart des propositions s'attaquent à un volet du problème, mais d'autres restent en suspend. Les études proposées à ce jour concernent deux principales catégories : méthodes de configuration avec détection de conflits et méthodes de configuration sans détection de conflits. Dans ce qui suit, nous allons décrire avec plus de détails chacune des deux catégories.

2.2.1 Autoconfiguration avec détection de conflits

Dans cette catégorie de méthodes, une adresse IP est proposée, ensuite un processus de détection de conflit est lancé sur tout le réseau pour s'assurer de son unicité. Dans le cas où le processus de détection échoue, une nouvelle adresse est choisie et le processus de détection est relancé. La différence entre les méthodes réside dans le choix de l'adresse IP de départ, dans le mécanisme de détection de conflits et dans l'aptitude de la méthode à traiter les problèmes relevant de la nature dynamique des réseaux ad hoc. A titre d'exemple, on peut citer les problèmes suivants : la gestion des départs, la récupération des adresses IP, le partitionnement et la fusion des réseaux.
Dans ce qui suit, nous allons présenter les méthodes s'inscrivant dans ce cadre, pour se faire une idée sur l'état de la recherche dans ce domaine.

a) Méthode de Perkins

Une première méthode faisant partie de cette catégorie est proposée par [Perkin 2001]. C'est la solution la plus simple, et elle a été proposée dans le cadre du groupe de travail Zeroconf[1] pour faire de l'autoconfiguration IPv4. Elle consiste à choisir aléatoirement une adresse parmi le bloc 169.254 / 16 [2], ensuite un mécanisme de détection d'adresses dupliquées est utilisé pour valider l'adresse proposée.

Le problème principal posé par cette méthode est qu'on ne peut pas garantir l'unicité de l'adresse. En effet, imaginons par exemple un nœud mobile qui quitte le réseau et entre temps un autre nœud qui arrive sur le réseau et qu'on lui attribue la même adresse du nœud partant. Un conflit d'adresse survient lorsque le nœud partant décide de regagner le réseau mère. Ce genre de scénario se reprend de façon générale lors d'un phénomène de fusion. Ainsi, quand deux réseaux ou plus fusionnent, il se peut que plusieurs nœuds se retrouvent avec les mêmes adresses IP. En plus, dans un cadre de déploiement mondial des MANETs, certains reprochent à cette méthode la limitation du nombre d'adresses IP. [Soo 2001] a tenté d'étendre la plage des adresses en proposant l'utilisation d'IPV6. Cependant, ceci ne résout pas le problème de fond qui est l'unicité et la gestion des partitionnements et des fusions.

b) Méthode de Nesargi et Prakash

Une autre méthode basée sur le mécanisme de détection de conflits est proposée par [Nesargi 2002]. L'idée consiste à choisir une adresse IP, puis diffuser un message sur le réseau pour avoir la permission de tous les autres nœuds. Malgré les apparences, cette solution diffère de celle proposée par Perkins dans le choix de l'adresse, dans le mécanisme de détection de conflit et enfin dans le traitement des départs, du partitionnement et de la fusion. Pour comprendre d'avantage cette approche, nous allons en décrire brièvement les principaux points.

---

[1] Groupe relevant du l'IETF, il travaille particulièrement sur les mécanismes d'autoconfiguration
[2] Ce bloque est enregistré auprès de l'IANA (Internet Assigned Numbers Authority)

- **Initialisation**

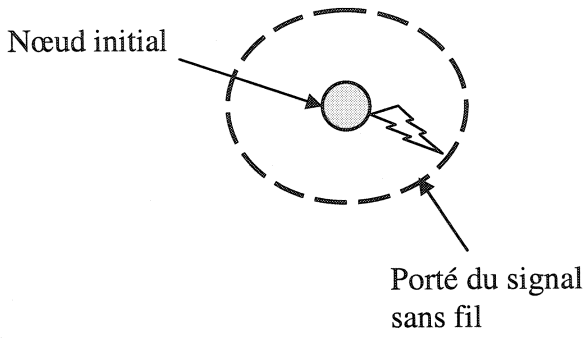


Figure 2.3 : Réseau de départ

Au départ, le premier nœud (figure 2.3) diffuse une demande de configuration et lance un compteur (Timer). Si après l'expiration du compteur, le nœud ne reçoit pas de réponse, il répète ce processus trois fois. Si toutes les tentatives échouent, il conclut qu'il est le premier nœud du réseau et s'autoattribue une adresse IP (la méthode suppose que chaque nœud est en mesure de choisir une adresse IP).

- **Arrivée d'un nouveau nœud**

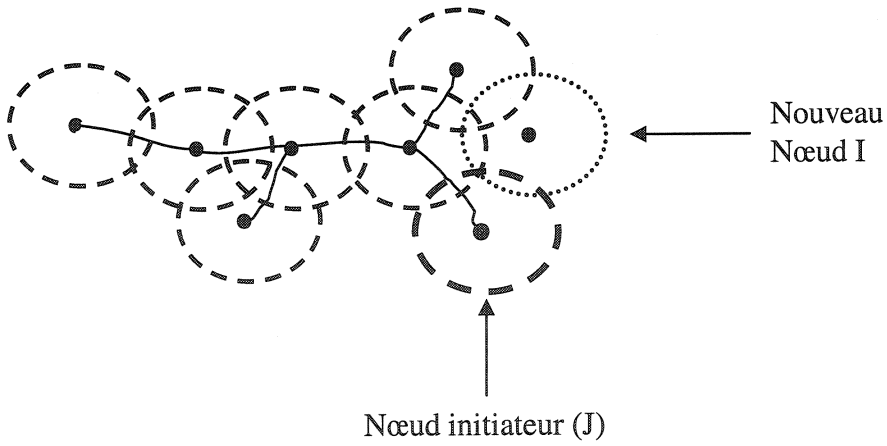


Figure 2.4 : Arrivé d'un nouveau nœud

Lorsqu'un nouveau nœud (nœud I) veut intégrer un réseau déjà en fonction, il diffuse une demande de configuration via son interface radio. Dans ce cas, il va recevoir plus

qu'une réponse car il existe déjà d'autres nœuds dans le réseau. Il choisit un des répondants comme initiateur et ignore les autres. Désignons par J l'initiateur sélectionné (figure 2.4). Chaque nœud K gère deux structures de données :

Allocated k : Contient toutes les adresses déjà en utilisation dans le réseau.

Allocate_Pending k : Contient toutes les adresses en cours d'allocation dans le réseau; cette structure est formée de tuples (adresse, initiateur).

Le nœud J choisit ensuite, une adresse X après avoir vérifié dans ses tables si celle-ci n'est pas affectée ni en cours d'affectation. Puis, il diffuse une demande de permission dans le réseau. Chaque nœud K recevant le message de J répond par l'affirmative si l'adresse X ne figure dans aucune de ses tables (Allocated k et Allocate_Pending k).

Si un autre nœud est en train d'attribuer la même adresse, il envoie à J une réponse négative et le processus de configuration est relancé avec une autre adresse. En revanche, si toutes les réponses sont affirmatives, l'adresse X est attribuée au nouveau noeud. Ensuite un message est diffusé sur tout le réseau pour informer les autres nœuds qu'une nouvelle adresse a été attribuée. Les autres nœuds notent cette adresse comme déjà allouée dans leurs tables respectives.

- **Départ volontaire**

On appelle départ volontaire ou normal, un départ où un nœud décide de quitter volontairement le réseau. Avant de partir, ce nœud diffuse un message informant tout le réseau de son départ. Ainsi une mise à jour des tables d'adresses sera accomplie par tous les nœuds.

- **Départ involontaire**

Ceci peut arriver lorsqu'un nœud décide de quitter soudainement le réseau. Cette méthode permet de détecter ce genre de départ au cours d'un processus de configuration. En effet, l'initiateur (qui s'occupe de la configuration d'un nouveau noeud) diffuse son message de demande de validation d'une nouvelle adresse IP. A priori, le nœud initiateur est censé recevoir une réponse de tous les nœuds du réseau. Si ce n'est pas le cas, le nœud initiateur tente de recontacter les nœuds qui n'ont pas répondu à trois reprises. S'il

n'y a toujours pas de réponse, il conclut qu'un départ est survenu, et diffuse un message pour en informer le reste du réseau.

- **Partitionnement et fusion**

Dans ce cas, la gestion du partitionnement du réseau est relativement simple à gérer car au bout d'un certain temps, chaque partition se rendra compte de l'absence de l'autre, ainsi des mises à jour des tables d'adresses seront effectuées. La fusion nécessite que chaque partition (sous-réseau) ait un identificateur. Ce qui fait que si deux partitions fusionnent, des informations de configuration seront échangées entre les nœuds et un nouveau réseau sera constitué.

Cette méthode apporte des solutions à la majorité des problèmes qui peuvent arriver pendant un processus de configuration. Les problèmes en question concernent la gestion des départs des nœuds, le partitionnement et la fusion.

Cependant, on reproche à cette méthode une performance assez limitée qui se manifeste essentiellement dans les points suivants :

- Surcharge du réseau : en effet, pour accomplir un processus de configuration, cette méthode a recours essentiellement à la diffusion. Ceci génère un trafic additionnel qui prend de l'ampleur dépendamment de la taille du réseau. En conséquence, les ressources (bande passante, énergie…) du réseau seront trop sollicitées juste pour des fins de configuration.
- Délai de configuration : en effet, pour qu'une adresse soit validée, la méthode citée plus haut doit attendre la réponse de tous les composants du réseau. Cette façon de faire entraîne des temps d'attente qui deviennent de plus en plus importants selon la taille du réseau.

c) Méthode de Weniger

Une autre approche s'inscrivant dans le cadre des méthodes de configuration avec détection de conflits est proposée par [Weniger 2002]. Weniger veut élargir le principe d'autoconfiguration d'IPv6 [Nilsen 2001] aux réseaux ad hoc. Pour se faire, des changements doivent être apportés à ce principe. En fait, le processus

d'autoconfiguration d'adresses IPv6 dans les réseaux filaires se décompose en trois étapes :

- La première étape consiste à créer une adresse lien-local pour la machine, c'est à dire une adresse valable uniquement sur le lien où celle-ci est située.
  Cette adresse est définie à partir de l'EUI-64 (Extended Universal Identifier) qui est à son tour définie à partir de la MAC adresse.
- La seconde étape consiste à vérifier l'unicité de l'adresse lien-local. Si cette adresse n'est pas unique, la procédure d'autoconfiguration s'interrompt et une intervention manuelle s'avère nécessaire.
- La troisième et dernière étape consiste à déterminer une adresse unicast globale par l'une des méthodes suivantes : autoconfiguration sans état, si l'attribution des adresses n'est pas gérée administrativement sur le site, ou autoconfiguration avec état dans le cas contraire.

Afin d'étendre l'autoconfiguration IPv6 au cas des réseaux ad hoc, cette approche propose de décomposer le réseau ad hoc en régions. Comme l'illustre la figure 2.5, chaque région dispose d'un représentant (leader node) élus selon une méthode expliquée dans [Weniger 2002]. Au niveau de chaque région, chaque nœud construit son adresse lien-local, ensuite chaque représentant choisit un identificateur de réseau, qu'il fait valider par un mécanisme de détection de conflit auprès des autres représentants de régions. Ainsi, chaque nœud dispose d'une adresse lien-local et d'un identificateur de réseau, et peut donc former sa propre adresse.

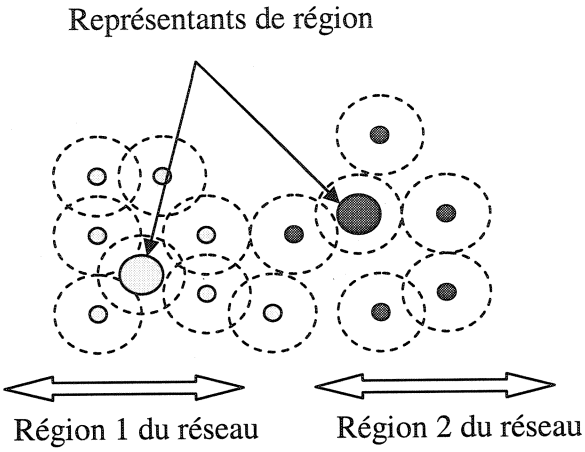


Figure 2.5 : Réseau multi-région

Le principal inconvénient de cette méthode c'est qu'elle utilise uniquement l'adressage IPv6, alors que par définition un réseau ad hoc peut être formé de n'importe quel composant indépendamment du fait qu'il soit configurable avec IPv4 ou IPv6. En plus la méthode susmentionnée ne tient pas compte des problèmes de partitionnements et de fusions qui sont courant dans un milieu aléatoirement mobile tel que les réseaux ad hoc. En outre, l'auteur précise que pour avoir de bons résultats, cette méthode doit être implantée avec un certain protocole de routage, ce qui restreint le champs de son application. On note aussi, le risque de conflit qui peut survenir pour les adresses lien-local, car cette dernière est calculée à partir de la MAC adresse. Or, l'unicité de la MAC adresse n'est pas souvent garantie à cause de la présence de produits de contrefaçon sur le marché.

2.3.2 Méthodes d'autoconfiguration sans détection de conflits

Les méthodes d'autoconfiguration faisant partie de cette catégorie n'ont pas besoin de mécanisme de détection de conflits car l'unicité de l'adresse est garantie à l'avance. Elles se basent par contre, pour certaines d'entre elles, sur la division binaire [Wilson 1995] des blocs d'adresses.

Parmi le peu d'approches se basant sur le principe d'allocation d'adresses sans détection de conflits, on trouve :

a) Méthode de Misra

Cette méthode de configuration est proposée par [Misra 2001] et porte le nom de DCDP (Dynamic Configuration and Distribution Protocol). Elle est basée sur le protocole DRCP (Dynamic Registration and Configuration Protocol) proposé par [McAuley 2000]. Le protocole DCDP est considéré comme une extension de DHCP (Dynamic Host Configuration Protocol) pour les environnements distribués et mobiles. Afin de comprendre cette méthode, nous allons en faire une brève description et on commence par le rôle des serveurs DCDP et DRCP :

- Les serveurs DCDP fournissent aux serveurs DRCP les informations de configuration.
- Les serveurs DRCP, effectuent des configurations en local moyennant des bloques d'adresses reçus au préalable d'un serveur DCDP.

Nous allons maintenant présenter un exemple d'illustration pour cette méthode.

Soit la topologie de la figure 2.6. On suppose qu'au départ, le serveur DCDP dispose du bloc d'adresses **192.1.1.0 – 192.1.18.255**.

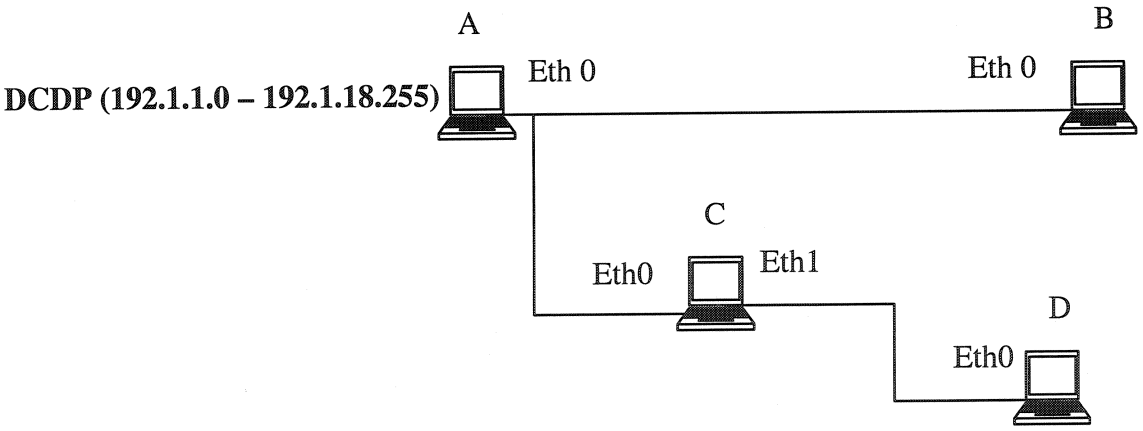


Figure 2.6 : Exemple de réseau avant la configuration

Au départ, tous les nœuds essaient de configurer leurs interfaces, mais sans succès car il n'y a pas de serveurs DRCP. Le DRCP inspecte auprès de chaque nœud et lance un appel DCDP pour obtenir des informations de configuration. Cet appel peut être voué à l'échec s'il y a absence de serveur DCDP dans l'entourage. Dans notre exemple, on a

un serveur DCDP au niveau du nœud A. Ce serveur répond à l'appel du DRCP de A et lui attribue les adresses 192.1.1.0/255. Ce dernier commence la configuration locale et attribue à l'interface Eth0 du noeud A l'adresse 192.1.1.1, à celle du noeud B l'adresse 192.1.1.2 et à celle du nœud C l'adresse 192.1.1.3. En revanche, l'interface Eth1 du nœud C reste sans configuration puisqu'il n'existe pas de serveur DRCP dans son entourage et le DRCP du nœud A ne peut pas accomplir cette tâche car l'interface Eth1 de C ne fait pas partie de son lien local. Le nœud C sollicite donc le DCDP du nœud A. Ce dernier effectue une division binaire de son bloc d'adresses et attribue au DCDP du nœud C le bloque 192.1.10.0 – 192.1.18.255. Le nouveau DCDP du nœud C alloue au DRCP du même nœud les adresses 192.1.10.0/255. Ainsi le DRCP de C est en mesure de configurer l'interface Eth1 de C et Eth0 de D comme indiqué dans la figure 2.7.

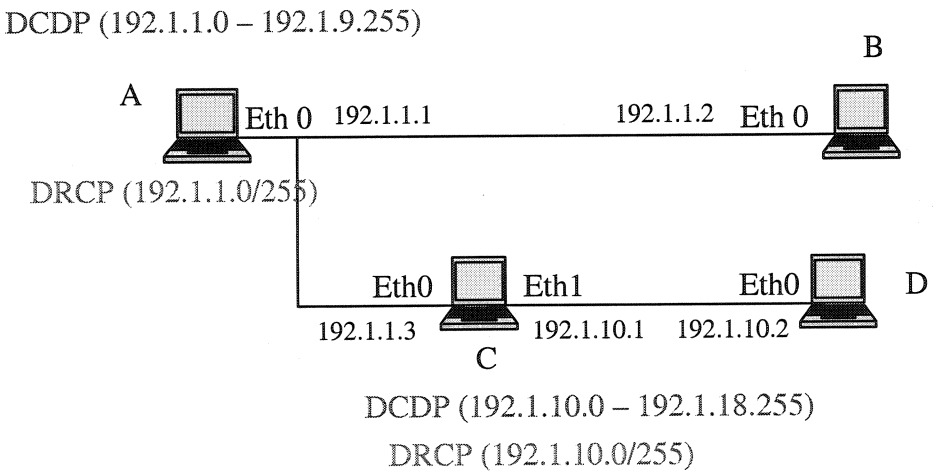


Figure 2.7 : Exemple de configuration avec DCDP et DRCP

Cette méthode présente l'avantage de faire une configuration graduelle en ajoutant au besoin des serveurs DCDP et DRCP. Cependant, rien ne garantit la présence d'un serveur DCDP ou DRCP de façon permanente dans le réseau à cause des départs qui peuvent survenir dans le réseau. En plus, l'utilisation des serveurs DCDP et DRCP suppose une certaine stabilité dans le réseau ce qui est loin d'être le cas suite au caractère très dynamique des réseaux ad hoc. En outre, ladite méthode ne gère en aucun cas les problèmes liés au partitionnement et à la fusion.

b) Méthode de Mohsin

Une autre méthode d'allocation d'adresses sans détection de conflits est proposée par [Mohsin 2002]. Cette méthode est basée elle aussi sur le principe de division binaire des blocs d'adresses [Wilson 1995]. Mais l'auteur a voulu que sa méthode évite les limitations de celles de Misra, en l'occurrence le problème de partitionnement et de fusion ainsi que la gestion des départs des nœuds. La méthode de Mohsin permet à chaque nœud de configurer d'autres, ainsi collectivement les nœuds réalisent la fonction d'un serveur DHCP. Plus précisément, le mécanisme de configuration d'un nouveau nœud se résume comme suit :

Chaque nœud est supposé gérer un ou plusieurs blocs d'adresses IP.

1- Un client (nouveau nœud) qui veut se joindre au réseau, diffuse sa demande via son interface sans fil.
2- Quand un serveur (un nœud du réseau) reçoit cette demande, il répond en utilisant son adresse IP et la MAC adresse du client. Il se peut que plusieurs serveurs répondent au même message; dans ce cas, un serveur est choisi, les autres sont ignorés.
3- Le client répond par un acquittement au serveur choisi.
4- Lorsque le serveur reçoit la réponse du client, il réalise qu'il est prêt pour un nouveau processus de configuration. Si le serveur a déjà des blocs d'adresses libres (suite à des départs par exemple), il attribue au client un de ses blocs. Sinon il lui donne la moitié de son propre bloc, ainsi qu'une copie de la table d'adresses.
5- Quand le client reçoit son bloc d'adresse, il se configure lui-même avec la première adresse de ce bloc et renvoie un message de confirmation au serveur en lui signalant que la configuration a réussi.
6- Quand le serveur reçoit cette confirmation il termine le processus de configuration.
7- Si un serveur n'a plus de blocs d'adresses disponibles, il contacte un de ses voisins pour s'en procurer de nouveaux.

- **Traitement des départs**

La gestion des départs des nœuds est d'une extrême importance dans cette méthode car chaque nœud a en sa possession des adresses libres. Donc s'il n'y a pas de récupération de cette ressource, le processus d'autoconfiguration s'arrêtera au bout de quelques itérations. C'est pourquoi un mécanisme de gestion des départs est mis en place.

- Départs involontaires

Ceci peut arriver lors d'un phénomène de partitionnement ou si un nœud décide soudainement de quitter le réseau. Ce genre de situation est détecté pendant les échanges périodiques d'informations de synchronisation. En fait, cette méthode impose à chaque nœud de contrôler le nœud disposant du bloc d'adresses complémentaire au sien. Ce contrôle se fait par un échange périodique et mutuel de messages. Si un nœud ne répond pas à son homologue, celui ci conclut que le nœud correspondant n'appartient plus au réseau mère. En conséquence, il considère son bloc d'adresses comme libre et l'annexe à son propre bloc.

- Départs volontaires

Dans ce cas, le nœud partant passe son bloc d'adresses à l'un de ses voisins. Ce dernier peut garder ce bloc d'adresses pour son compte ou chercher le nœud disposant du bloc complémentaire (l'autre moitié de ce bloc) et lui fournir ensuite cet espace d'adresses.

- **Traitement du partitionnement et de la fusion**

Dans cette méthode, le partitionnement du réseau se réduit avec le temps à un problème de départ. En effet, pendant l'échange des messages de synchronisation, chaque nœud peut détecter l'absence du nœud qui détient le complémentaire de son bloc d'adresses. Et, ainsi avec le temps, les parties qui ne font plus partie du réseau mère seront détectées. Par contre la fusion est moins évidente à résoudre. La méthode propose l'utilisation d'identificateur de réseau. Chaque partition calcule son identificateur et par

la suite si une nouvelle fusion arrive les nœuds ont recours aux identificateurs pour résoudre les éventuels conflits d'adresses.

Cette méthode permet une allocation directe d'adresses IP, traite les problèmes de départs volontaires et involontaires ainsi que ceux liées au partitionnement et à la fusion. Cependant, la méthode susmentionnée présente les faiblesses suivantes :

- Le mécanisme proposé pour gérer les départs présente de sérieuses limites surtout lorsqu'il s'agit des départs involontaires. En effet, cette gestion se base sur le contrôle mutuel que doivent entretenir les nœuds ayant des blocs d'adresses complémentaires. Le problème survient lorsque les deux nœuds en question quittent le réseau sans préavis : comment les deux blocs correspondant vont-ils être récupérés ? Et, avec le temps, les blocs d'adresses devant subir de plus en plus de division, comment doit être réaliser le contrôle dont on a parlé précédemment ?
- Concernant l'étude de la fusion proposée dans cette méthode, nous estimons qu'elle est incomplète. Ceci est dû au fait que le problème de fusion est abordé avec une idée d'identificateur de réseau, l'incomplétude à ce niveau résulte du fait que la méthode ne spécifie pas qui va calculer l'identificateur du réseau.

c) Méthode de Zhou

La méthode de [Zhou 2003] se base sur l'idée qu'un nœud peut construire une adresse IP à partir de la sienne. Cette fois, la technique de division de blocs d'adresses, comme c'était le cas avec les méthodes de [Mohsin 2002] et de [Misra 2001], n'est plus utilisée. Mais, une fonction $f()$ est conçue pour fournir une nouvelle adresse IP à partir de celle du nœud courant. Pour qu'un même nœud soit en mesure de fournir plusieurs adresses IP différentes à partir de son adresse, une variable d'état est ajoutée comme argument de la fonction $f()$. Les étapes de configuration d'un nouveau nœud sont résumées comme suit :

Soit A un nœud initial et B un nouveau qui demande une adresse.

1- Le premier nœud du réseau, qu'on note A, choisit son adresse IP de façon aléatoire : *address_A*. Dans ce cas, *address_A* est considérée aussi comme valeur de sa variable d'état notée *Stat_A*.
2- Lorsqu'un nouveau nœud, noté B, demande une adresse à A, celui ci calcule *address_B* = f(*address_A*). Ensuite, il fournit cette adresse, avec une variable d'état à B et met à jour sa propre variable d'état.
3- Le nœud B utilise donc *address_B,* comme son adresse IP, et la valeur de sa variable d'état est celle fournie par A.
4- Les nœuds A et B deviennent donc prêts à accomplir de nouvelles configurations.

Cette méthode repose donc sur la définition de la fonction $f()$ et sur la génération de différentes variables d'états. Dans la conception de $f()$, on suppose qu'on peut:

- Obtenir une nouvelle valeur de $f()$, à chaque fois qu'on change de variable d'état.
- La probabilité pour qu'on puisse générer pour deux variables d'états différentes une même valeur de $f()$ est minime.

L'exemple concret qu'utilise [Zhou 2003] dans sa méthode est le suivant :

- Tout d'abord, la génération des variables d'états se base sur un résultat connu de l'arithmétique qui consiste à écrire n'importe quel nombre entier sous une forme canonique. Soit un nombre entier, noté N, N peut s'écrire sous forme d'un produit de nombres premiers, $N = \prod_{i=1}^{k} P_i^{ei}$, avec les P_i satisfont :

  $\begin{cases} P_1<P_2 \ldots<P_k \\ e_i \text{ des exposants entier non négatifs, } 1 \leq i \leq k \end{cases}$

  Donc avec des k-tuples (e_1, e_2,…,e_k), on peut avoir des entiers différents, et par la suite des variables d'états différentes.
- d'autre part, pour la conception de $f()$, on pose k=4 et une nouvelle adresse est obtenue par : address = $(a + 2^{e_1} + 3^{e_2} + 5^{e_3} + 7^{e_4}) \bmod range + 1$ avec range

suffisamment grand pour rendre minime la probabilité de générer des adresses dupliquée, et "a" représente l'adresse du nœud initiale du réseau. Un nœud est représenté alors par (address, (e_1, e_2, e_3, e_4)).

L'exemple de la figure 2.8, illustre comment cette méthode se comporte pour répondre à des demandes de configurations.

1- Le premier noeud génère une adresse aléatoire noté : a, et commence avec l'état (0,0,0,0).
2- Lorsque B demande une adresse, A lui transfère l'adresse initiale "a" et la liste des exposants e_i, qui vaut dans ce cas : (0,0,0,0).
3- B reçoit l'adresse "a", incrémente l'exposant souligné, et déplace le soulignement sur vers la droite. Elle se retrouve alors avec une adresse qui vaut " a+3 " et une liste d'exposants égale à : (1,0,0,0).

On note que dans cette méthode toutes les adresses se déduisent à partir de celle du premier nœud, en l'occurrence " a ". L'incrémentation de l'indice souligné se fait à chaque nouvelle configuration, et le déplacement du soulignement se fait uniquement chez le nouveau nœud.

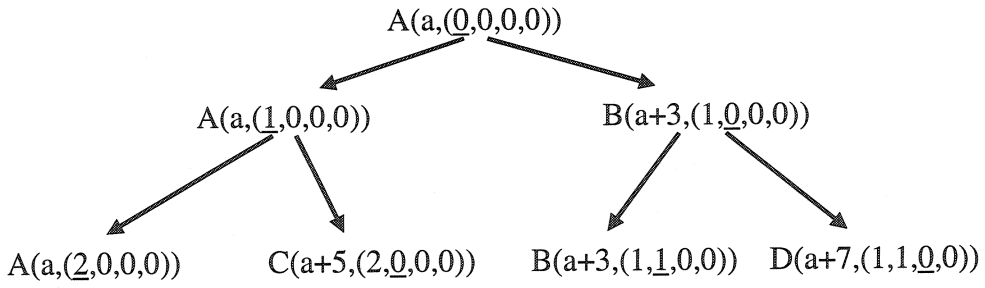


Figure 2.8 : Exemple de configuration avec la méthode de Zhou

La méthode présentée plus haut, propose une nouvelle approche de configuration basée sur une fonction conçue spécialement à cet effet. Elle permet d'accomplir des

configurations de façon directe et progressive. Le problème, c'est qu'on ne peut pas garantir l'unicité des adresses de façon indéfinie. En effet, dans la pratique, le "range" définit précédemment ne peut pas dépasser une limite donnée, et avec les incrémentations progressives des exposants e_i, on peut atteindre facilement la limite du "range" et compromettre alors l'unicité des futurs adresses proposées. En plus, aucun mécanisme de récupération d'adresses des nœuds qui ne font plus partie du réseau n'est considéré. Le fait de ne pas prévoir de mécanisme de récupération d'adresses, conduira, avec le temps, à un épuisement des adresses IP disponibles.

2.3.3 Récapitulatif

Durant cette partie, nous avons présenté les résultats de notre recherche bibliographique relatifs au problème d'autoconfiguration dans les réseaux mobiles ad hoc. Les méthodes d'autoconfiguration qu'on a pu énumérer dans la littérature étaient soit des méthodes avec détection de conflits ou des méthodes sans détection de conflits. Les méthodes d'autoconfiguration avec détection de conflits (ou avec détection d'adresse dupliquée) consistent à proposer une adresse IP et ensuite à voir s'il y a consensus de tous les nœuds du réseau sur ce choix pour assurer l'unicité; par contre, avec les méthodes sans détection de conflits, l'unicité de l'adresse est préalablement garantie. Dans ce cas, un mécanisme de division binaire d'adresses ou une fonction de configuration s'avère nécessaire. Les méthodes d'autoconfiguration avec détection de conflit ont l'avantage d'être simple et évolutive. Cependant, elles présentent des faiblesses qui rendent leur efficacité discutable. Par exemple, l'utilisation de la diffusion génère un important trafic supplémentaire, ce qui épuise les ressources du réseau juste pour des fins de configuration. En plus, la perte de messages peut occasionner des conflits d'adresses, ce qui est intolérable pour tout échange dans le réseau. La diffusion entraîne des délais de configuration relativement longs, car il faut attendre le résultat du mécanisme de détection d'adresse dupliquée avant d'attribuer une adresse de façon définitive. Par contre, les méthodes d'autoconfiguration sans détection de conflits assurent une allocation d'adresse directe et, en principe, sans attente. Ils utilisent donc moins de diffusion et l'unicité des adresses est garantie à l'avance, de façon définitive

avec certaines méthodes et sous conditions avec d'autres. L'inconvénient de ces méthodes réside dans les pertes de blocs d'adresses (principale ressource) qui peuvent survenir, même si certains méthodes intègrent des mécanismes de récupération d'adresses. En plus, le fait de répartir la configuration sur tout le réseau, rend la gestion des départs peu efficace. Afin d'avoir une vision globale sur l'état des méthodes de configuration discutées jusqu'à présent, on propose le tableau récapitulatif suivant :

Tableau 2.1 : Récapitulatif des méthodes d'autoconfiguration pour MANETs

Méthode de configuration	Unicité de l'adresse	Diffusion	Problèmes de partitionnement et de fusion	Mécanisme de récupération des adresses	Complexité
Perkins	non garantie	oui	non traité	non	simple
Nesargi & Prakash	garantie sous condition	oui	traité	pas trop développé	moyenne
Weniger	garantie localement	oui	non traité	non	moyenne
Misra	garantie localement	faible	non traité	non	moyenne
Mohsin	garantie	faible	traité	non	moyenne
Zhou	garantie sous condition	faible	traité	non	assez

Dans ce qui suit, notre but sera de proposer une nouvelle solution d'autoconfiguration pour les réseaux mobiles ad hoc en gardant au maximum les avantages des présentes méthodes et en évitant le plus possible leurs limitations.

Chapitre 3
Protocole d'autoconfiguration pour MANETs (APM)

Nous avons précédemment présenté le problème de configuration dans les réseaux mobiles ad hoc. Ce problème consistait à trouver un mécanisme d'attribution d'adresses IP pour les unités mobiles qui veulent intégrer un réseau déjà en fonction. La difficulté de ce problème réside dans le caractère aléatoire des nœuds mobiles. Jusqu'à présent quelques solutions ont été proposées. Certaines d'entre elles effectuent une configuration avec détection de conflits. Tandis que d'autres méthodes, accomplissent des configurations sans détection de conflits. Notre objectif dans ce travail est de proposer une nouvelle approche de configuration pour les réseaux ad hoc. Celle ci sera axée sur une allocation directe d'adresses IP et sur une gestion centralisée des problèmes liés aux comportements à la fois dynamiques et aléatoires des MANETs. Dans le présent chapitre, nous présenterons dans un premier temps les exigences du mécanisme de configuration que nous allons proposer. Ensuite nous annoncerons les hypothèses de base qui seront prise en compte dans ce travail. Enfin, nous donnerons une description détaillée de notre protocole de configuration.

3.1 Exigences du protocole

L'approche que nous allons proposer doit respecter certaines exigences. Ces exigences représentent une grande partie des atouts des méthodes d'autoconfiguration précédemment présentées, ainsi que nos propres exigences sur le futur protocole. On entend dire par exigence, les points favorables que nous jugeons indispensables pour une configuration efficace. Parmi ces points, on peut citer par exemple, les problèmes de fusion et de partitionnement. Ces problèmes ont été traités dans les propositions de

configuration de [Nesargi 2002] et [Mohsin 2002]. Nous allons nous aussi les adresser mais différemment.

Les points essentiels auxquels notre protocole doit répondre se résument comme suit :

1. Absence de conflits d'adresses : ceci veut dire qu'à tout instant, chaque nœud du réseau ad hoc utilisera une et une seule adresse IP pour communiquer.
2. Une adresse IP est utilisée uniquement à l'intérieur du réseau : ceci signifie que lorsqu'un nœud quitte le réseau, son adresse IP doit être récupérée.
3. Un nœud peut effectuer plusieurs configurations à la fois. C'est à dire que des processus de configuration peuvent être accomplis de façon simultanée.
4. La configuration se fait de façon directe : autrement dit, l'avis des autres nœuds du réseau n'est pas sollicité.
5. L'évolutivité du réseau est garantie : ceci veut dire qu'il n'y a aucune restriction sur les nouveaux nœuds qui veulent intégrer un réseau ad hoc opérationnel.
6. Les problèmes de partitionnement et de fusion sont traités adéquatement.
7. Un mécanisme de gestion des départs et de récupération d'adresses IP est prévu.
8. Chaque nœud est libre de quitter le réseau à tout instant.

3.2 Hypothèses

Pour rester dans le cadre du sujet traité, le protocole que nous allons proposer tiendra compte de certaines hypothèses. Ces hypothèses concernent essentiellement la sécurité et la disponibilité d'un bloc d'adresses de départ. Nos principales hypothèses sont :

Hypothèse 1 : On considère que le bloc d'adresses IP à partir duquel les unités mobiles seront configurées est connu à l'avance par tous les nœuds du réseau.

Hypothèse 2 : On suppose qu'on n'a pas de nœuds malicieux dans le réseau. C'est-à-dire que les nœuds ne perturbent en aucun cas le fonctionnement normal du réseau. Cette hypothèse va nous permettre de nous pencher sur l'élaboration d'un protocole sans contraintes de sécurité. Les aspects reliés à la sécurité seront pris en compte dans des travaux futurs.

Hypothèse 3 : Les unités mobiles sont supposées avoir de bonnes capacités de calcul et de mémoire. En conséquence, n'importe quelle unité mobile est en mesure d'assurer le service de configuration au sein du réseau.

3.3 Notations

Dans cette partie, nous décrirons toutes les structures, les abréviations et les mots qui vont figurer dans la description de notre protocole.

- **Nc** : désigne le Nœud de Configuration du réseau. Il s'agit d'un nœud particulier qui se charge de la gestion des problèmes liés au réseau. Il effectue aussi un contrôle de l'état du réseau. Ce contrôle consiste à faire, par exemple, le suivi du trafic ou de la taille du réseau. Cette façon de faire lui permet une meilleure gestion du service de configuration. Plus de précisions sur le fonctionnement du Nc vont être données tout au long de ce chapitre.
- **TA** : désigne une structure appelée Table d'Adresse qui permet au Nc d'avoir une vision générale sur les adresses IP du réseau. Cette structure est constituée de deux champs : adresses_libres et adresses_allouées. Le champ adresses_libres désigne les adresses IP non encore utilisées dans le réseau et le champ adresses_allouées réfère aux adresses IP en cours d'utilisation.

  TA
  - **adresses_libres**
  - **adresses_allouées**

- **Id_Network** : ce champ caractérise le réseau, il change à chaque fois qu'un nouveau Nc est choisi, ou suite à une mise à jour à l'échelle du réseau de la table d'adresses. Son calcul peut se faire, par exemple, à partir de l'adresse MAC du Nc courrant, de celle de ses voisins et d'une valeur générée de façon aléatoire. Ceci permet de garantir l'unicité de ce paramètre.
- **Configuration_Address** : ensemble d'adresses IP libres que possède un nœud donné.

- **Confirm_Config** : message de confirmation envoyé à l'accompagnateur. Ce dernier fait référence au nœud qui va assister un nouveau composant mobile pour avoir une adresse IP.
- **Address_Request** : réfère à une demande d'adresse envoyée par un nouveau nœud à ses voisins immédiats.
- **Address_Reply** : réponse à une demande d'adresse, envoyée par un nouveau nœud.
- **Config_Request** : demande de configuration envoyée par un accompagnateur au nœud de configuration.
- **Compt_Config** : compteur de configuration.
- **New_Nc** : message pour annoncer qu'un nouveau Nc est choisi, il contient aussi une information sur l'identificateur (Id_Network).
- **Next_Nc** : Message pour informer un nœud qu'il a été choisi pour devenir le prochain nœud de configuration.
- **Next_Nc_Reply** : Message de réponse, envoyé suite à la réception d'un message Next_Nc.
- **Update_Request** : message de demande de mise à jour.
- **Update_Reply** : réponse à une demande de mise à jour.

3.4 Principe de fonctionnement

Soit un réseau mobile ad hoc comme c'est décrit dans la figure 3.1. Ce dernier est constitué d'unités mobiles et d'un nœud de configuration Nc. Une unité mobile communique avec le reste du réseau moyennant son interface radio et par le biais de l'adresse IP du destinataire. Quant au nœud de configuration, il réfère à une unité mobile qu'on a choisi pour offrir un service de configuration au reste du réseau. Ce service, consiste à fournir des adresses de configuration et à gérer les problèmes liés à la nature dynamique de ce genre de réseau. Le choix du nœud Nc se fait parmi les composants mobiles du réseau, plus de détails à ce sujet seront donnés ultérieurement. Le nœud Nc est unique sur le réseau, il gère une table d'adresses TA qui contient les adresses IP

libres et les adresses IP en utilisation (déjà allouées). Le noeud Nc reçoit les demandes d'adresses IP, et retourne une réponse contenant des adresses de configuration.

L'idée de base de notre protocole est de permettre à un nouveau nœud d'intégrer un réseau ad hoc déjà en fonction, en lui fournissant une adresse IP. Ceci est sujet à un processus d'attribution d'adresses IP que nous avons appelé : processus d'ajout d'un nouveau nœud au réseau. Ce processus se résume comme suit : quand un nouveau nœud veut rejoindre un réseau mobile ad hoc, il contacte (via son interface radio) un de ses voisins immédiats, et lui passe une demande de configuration. Ce voisin, que nous appellerons accompagnateur, lui fournit une adresse IP et l'identificateur courant du réseau. Dans le cas où l'accompagnateur possède des blocs d'adresses libres, appelée : adresses de configuration, il en fournit une partie au nouveau nœud. Dans le cas contraire, l'accompagnateur contacte le Nc pour chercher de nouvelles adresses de configuration.

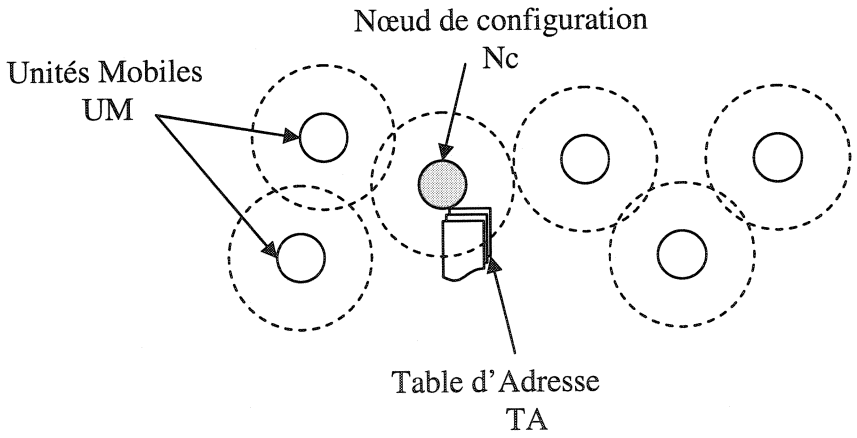


Figure 3.1 : Exemple de réseau mobile ad hoc avec nœud de configuration

3.5 Description du protocole

Dans cette partie nous allons décrire, étape par étape, notre proposition d'autoconfiguration pour les réseaux mobiles ad hoc que nous avons nommé : protocole d'autoconfiguration pour MANTEs ou APM (Autoconfiguration Protocol for MANETs). La première étape que nous allons discuter est celle de l'initialisation. Plus exactement,

nous allons montrer comment le réseau ad hoc sera constitué et configuré pour la première fois. Ensuite, nous présenterons le processus d'arriver d'un nouveau nœud dans le réseau. Après, nous traiterons le problème de départ de nœuds y compris le nœud de configuration (Nc). Enfin, nous terminerons par une proposition traitant les problèmes de partitionnement et de fusion.

3.5.1 Phase d'initialisation

Le premier nœud qui va constituer un réseau mobile ad hoc a besoin lui aussi de configuration. Pour se faire, on propose le processus suivant :

1- Le premier nœud envoie un message *Address_Request* et déclenche son compteur *Compt_Config*.
2- Si après expiration du compteur, aucune réponse n'est reçue, un autre message *Address_Request* est envoyé.
3- Si au bout de la troisième tentative il n'y a toujours pas de réponse, il conclut qu'il est le seul nœud du réseau, et s'autoattribue une adresse IP du bloc considéré dans l'hypothèse 1.
4- Ce nœud est considéré comme nœud de configuration, une première table d'adresse est mise à jour et un nouvel identificateur de réseau est calculé.

Algorithme

L'algorithme ci-dessous (figure 3.2) traduit la phase d'initialisation de notre approche de configuration. Il sera donné en notation APN (Abstract Protocol Notation).

```
begin
 const  Compt_Config = X // taille du compteur
 Address_Reply = null
 nombre_essai = 1
 Repeat
  broadcast Address_Request // diffusion d'une demande d'adresse
  while(Compt_Config !=0  and  Address_Reply = null)
   wait(t) // délai d'attente
```

```
 Compt_Config - - // décrementation du Compt_Config
end while
if(Address_Reply != null)
 apply(new_node_process) // appliquer le processus d'arriver
 break
end if
 nombre_essai ++  // incrémenter le nombre d'essai
until(nombre_essai=3)
if(nombre_essai=3)
 get_IP_adress() // le noeud s'auto attribue une adresse IP
 calculate(new_id) // calculer un nouveau identificateur
 update(TA) // mise à jour de la table d'adresse
end if
End
```

Figure 3.2 : Initialisation du réseau

Cette phase du protocole est la base de tout réseau ad hoc, elle ne nécessite aucune considération particulière à prendre en compte, sauf la connaissance préalable du bloque d'adresse à partir duquel les configurations vont être accomplis. Dans la figure 3.3, nous donnons une illustration schématique du processus d'initialisation cité plus haut.

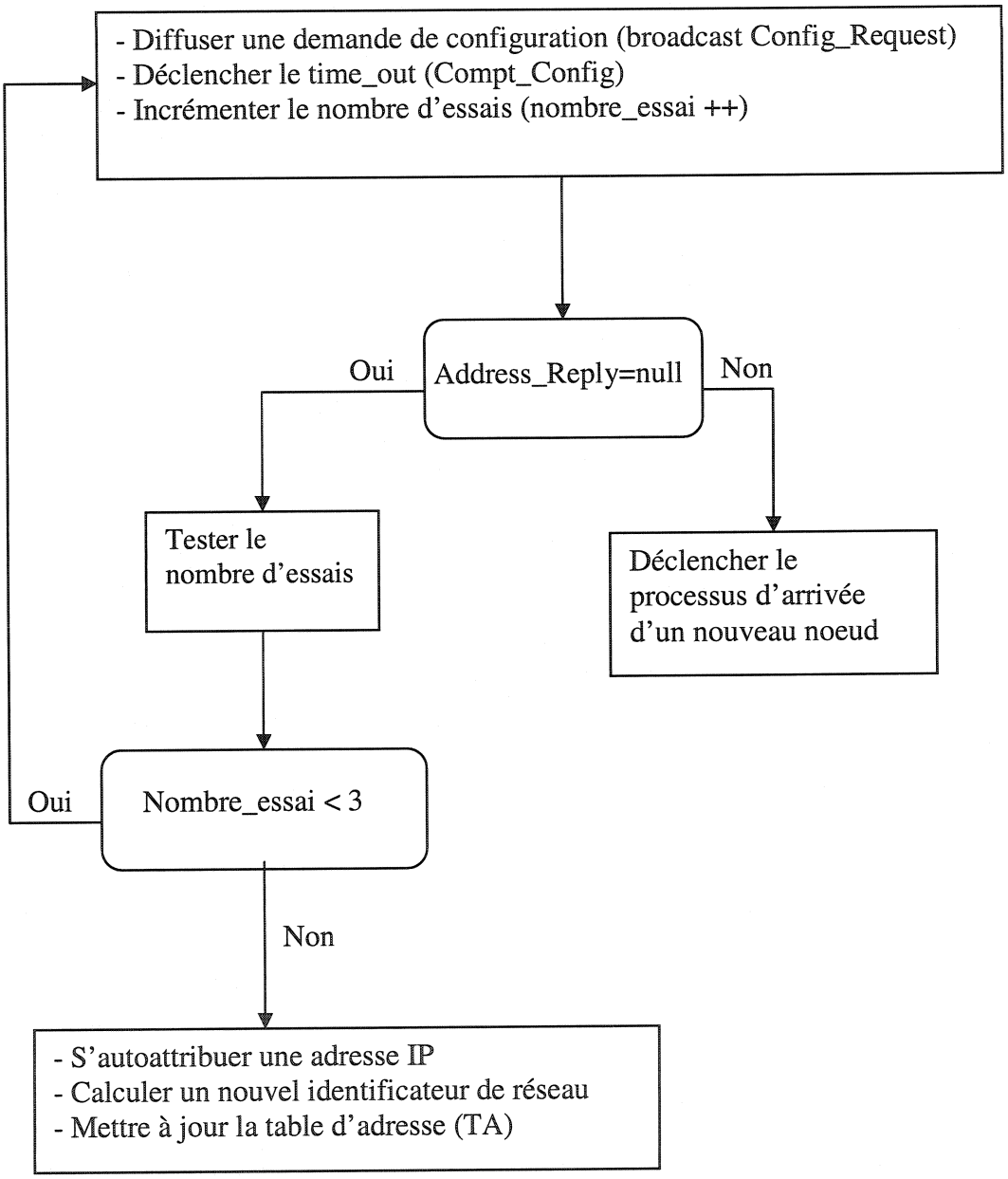


Figure 3.3 : Processus d'initialisation

Exemple :

Soit le bloc d'adresses suivant : 132.10.1.1 – 132.10.254.254.

Après avoir configuré le premier nœud, la table TA contiendra les informations suivantes :

adresses_libres = (132.10.1.2 – 132.10.254.254)

adresses_allouées = 132.10.1.1

3.5.2 Arrivée d'un nouveau nœud

Lorsqu'une unité mobile veut faire partie d'un réseau ad hoc déjà en fonction, elle a besoin d'une adresse IP pour pouvoir communiquer avec le reste du réseau. Pour se faire, elle envoie une demande d'adresse à ses voisins immédiats (Address_Request). Un ou plusieurs voisins peuvent répondre à cette demande. À ce moment, l'unité mobile choisit un des répondants comme accompagnateur et ignore les autres. Ici, nous avons adopté l'idée de l'accompagnateur proposée par [Nesargi 2002]. Le rôle de l'accompagnateur est d'assister le nouveau nœud pour devenir membre du réseau en lui fournissant une adresse IP. Le nouveau noeud envoie une confirmation (Confirm_Config) à l'accompagnateur choisi. Ce dernier fournit immédiatement au nouveau nœud, une adresse IP et l'identificateur courrant du réseau et éventuellement des adresses de configuration. Les adresses de configuration réfèrent à un ensemble d'adresses libres, généralement sous forme de blocs, que l'accompagnateur utilise pour satisfaire ses demandes de configuration. Si un accompagnateur possède suffisamment d'adresses de configuration, il les partage avec le nouveau nœud. Sinon, il envoie une demande de configuration au Nc pour solliciter d'autres adresses. Le Nc lui répond en envoyant des adresses de configuration, dont une partie est par la suite acheminées au nouveau nœud.

Dans la figure 3.4 nous présentons les différents échanges effectués entre l'accompagnateur et le nouveau nœud d'une part et entre l'accompagnateur et le nœud de configuration (Nc) d'autre part.

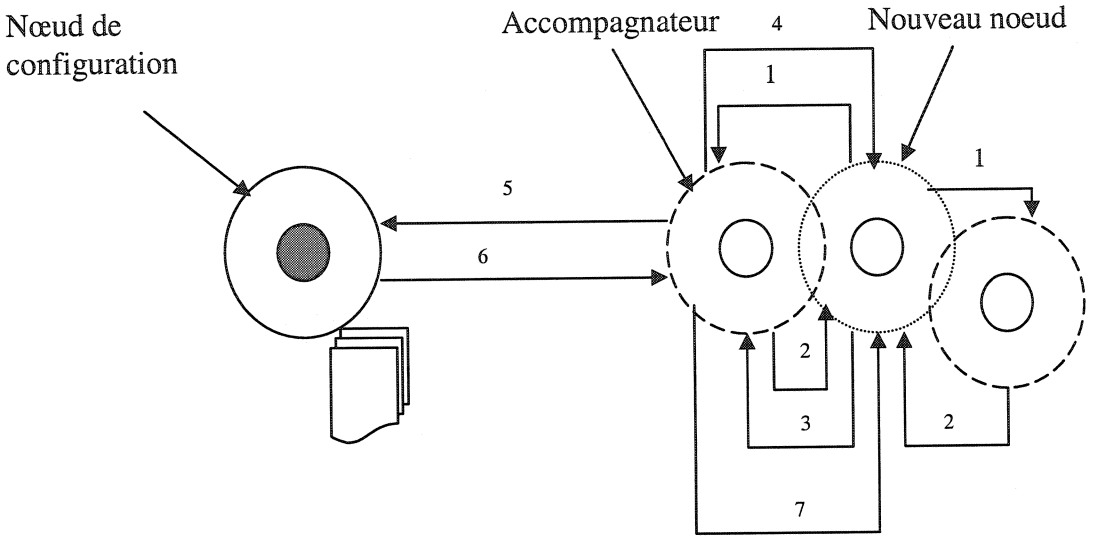


Figure 3.4 : Arrivée d'un nouveau nœud au réseau

1. Le nouveau nœud diffuse une demande d'adresses : *Address_Request* à ses voisins à travers son interface radio : ainsi, seuls les voisins entrant dans son champ de communication reçoivent cette requête.
2. Les nœuds voisins répondent par un *Address_Reply* au nouveau nœud.
3. Le nouveau nœud choisit un de ces voisins comme accompagnateur et ignore les autres, et déclenche son compteur de configuration : *Compt_Config*. Le choix de l'accompagnateur se fait par l'envoi d'un message de confirmation : *confirm_msg*.
4. L'accompagnateur choisi fournit immédiatement au nouveau nœud une adresse IP, l'identificateur de réseau et éventuellement des adresses de configuration (adresses libres).
5. Si l'accompagnateur ne possède plus d'adresses libres, il envoie au Nc une demande de configuration : *Config_Request*.

6. Le Nc répond à l'accompagnateur en lui envoyant des adresses de configuration sous forme de bloc, ensuite il met à jour sa table d'adresses. La mise à jour consiste à marquer les adresses envoyées comme étant déjà allouées.
7. L'accompagnateur transfert par la suite une partie des adresses de configuration au nouveau nœud.

Dans l'étape 3, il se peut que le compteur "Compt_Config" expire avant que le nouveau noeud ne reçoive d'adresse IP suite à une perte de messages par exemple. Dans ce cas, un nouveau processus de configuration sera lancé. Et si entre temps une réponse de configuration tardive arrive au nouveau nœud, celle-ci sera adoptée. Les autres réponses seront renvoyées à leurs expéditeurs. Il se peut aussi que le nœud demandeur de configuration quitte la zone de communication sans fil de l'accompagnateur avant que le processus de configuration ne soit terminé. Ce cas sera traité comme un départ involontaire dont les détails seront exposés ultérieurement. Le contact du nœud de configuration effectué à l'étape 5 peut en fait se faire dans deux cas de figure : le premier cas est justement celui que nous avons vu dans la description de cette étape, à savoir quand l'accompagnateur ne dispose plus d'adresses de configuration. Un autre cas est celui où l'accompagnateur ne possède aucune adresse libre pour répondre à une demande de configuration. Dans son choix d'accompagnateur, le nouveau noeud choisit en premier les voisins disposant d'adresses de configuration. Si aucun des voisins ne possède de blocs d'adresses libres, il choisit un des voisins qui peut lui fournir une adresse IP immédiatement. Dans le dernier cas, où aucun de ces voisins ne possède des adresses libres, il choisit un parmi ceux-ci comme accompagnateur. Le nœud qui sera choisi comme accompagnateur, dans ce cas, va chercher une adresse IP auprès du Nc. Ce genre de situation peut arriver lorsque le nœud de configuration ne possède plus assez d'adresses. Le rôle du Nc comme contrôleur du processus de configuration réside dans sa capacité à changer de stratégies de configuration dépendamment de la taille du réseau et aussi dépendamment de son actif en terme d'adresses libres. En résumé, notre protocole d'autoconfiguration APM, effectue une hiérarchisation du processus de configuration. En effet, le premier niveau de cette hiérarchie représente le cas où

l'accompagnateur possède suffisamment d'adresses de configuration. A ce moment la configuration est entièrement réalisée entre le nouveau nœud et son accompagnateur. L'avantage de ce niveau de configuration est que l'allocation d'adresses se fait de façon directe et sans délai d'attente (délai assez limité et dépendent principalement de l'interface air). En outre, la signalisation est assez limitée. Le deuxième niveau de configuration correspond au cas où l'accompagnateur fournit une adresse IP au nœud demandeur, mais doit contacter le Nc pour chercher des adresses de configuration, pour son compte et pour celui de son client. Dans ce cas, la configuration d'un nouveau nœud se fait toujours sans délai d'attente, mais elle génère un trafic additionnel sur le réseau. Le troisième niveau de cette hiérarchie, est le cas où l'accompagnateur est obligé de faire attendre le nouveau nœud, en allant lui chercher une adresse IP chez le Nc. Ce cas entraîne un délai d'attente et un trafic supplémentaire sur le réseau dû principalement aux échanges de messages entre l'accompagnateur et le Nc. On peut même imaginer un cas extrême où la configuration d'un nœud coïncide avec un départ de Nc. Le nouveau nœud doit alors attendre jusqu'à ce qu'un nouveau Nc soit choisi. Cette fois, le délai d'attente va inclure également le délai d'exécution du mécanisme de remplacement du Nc. Tous les cas cités plus haut seront revus lors de nos analyses de performance qui feront l'objet du chapitre 4.

La phase de configuration d'un nouveau nœud peut selon les cas se répéter plus qu'une fois. Ceci peut être due à des pertes de messages, à la mobilité du nœud demandeur ou au départ de l'accompagnateur. Le plus important c'est que notre protocole prévoit la continuité du processus de configuration. Et, même si le nombre d'essais est supérieur à trois par exemple, le nouveau nœud n'appliquerait pas la phase d'initialisation car il s'est rendu compte déjà de l'existence d'autres unités mobiles.
La représentation schématique de la figure 3.5, illustre d'avantage le processus d'ajout d'un nouveau nœud dans un réseau ad hoc en fonction.

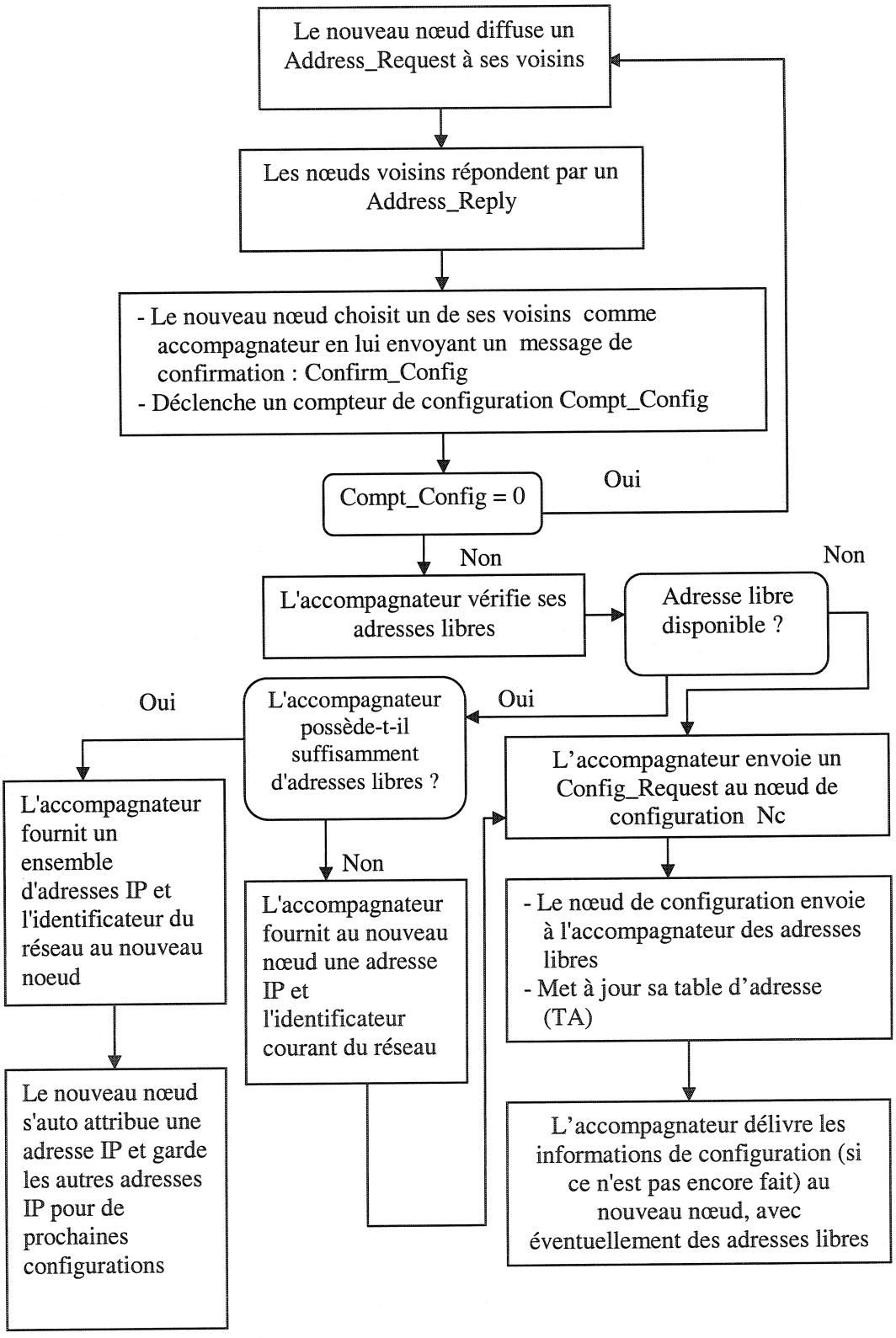


Figure 3.5 : Processus d'ajout d'un nouveau noeud

3.5.3 Gestion des départs des unités mobiles

On désigne par départ d'une unité mobile le fait que celle-ci s'éloigne du réseau de telle façon qu'aucune autre unité mobile du réseau ne puisse communiquer avec elle. Les départs concernent aussi bien le nœud de configuration que le reste des composants du réseau. La gestion des départs est d'une grande importance. En effet, lorsqu'il s'agit, par exemple, du départ du nœud de configuration, un mécanisme de remplacement doit être déclenché pour que le réseau puisse continuer à assurer le contrôle du service de configuration. Et, dans le cas du départ d'une unité mobile autre que le nœud Nc, l'adresse de celle-ci doit être récupérée en vue d'une prochaine réutilisation. Dans ce qui suit, nous allons traiter le problème des départs en tenant compte des différents cas possibles.

a) Départ d'une unité mobile

Vu leurs comportements aléatoires, les nœuds d'un réseau mobile ad hoc peuvent à tout moment quitter le réseau mère. Ceci peut survenir de façon volontaire ou involontaire. Un départ volontaire signifie qu'une unité mobile, pour une raison ou pour une autre, décide de quitter à son gré le réseau. On appelle aussi ce genre de départ : départ programmé. Quant au départ involontaire, il arrive de façon accidentelle et inattendue. Il peut arriver suite à un "crash", un déni de service ou de manière générale suite à un événement imprévu. Les départs des unités mobiles autres que le nœud de configuration seront détectés pendant la mise à jour de la table d'adresse du Nc, effectuée à la suite d'échange de messages Update_Reply. Ces messages sont échangés, par exemple, lors du remplacement du Nc, ou lorsqu'un phénomène de fusion est en cours de traitement. Ces derniers points, seront vus avec plus de précisions dans les sections à venir.

b) Départ du nœud de configuration

Le nœud de configuration peut lui aussi quitter le réseau de façon intentionnelle (volontaire), ou à l'improviste (involontaire). Dans les deux cas, un mécanisme de remplacement doit être déclenché.

- Cas d'un départ intentionnel

Dans ce cas le nœud de configuration programme son départ à l'avance. Cette programmation consiste à choisir un successeur (nouveau Nc) à partir de ces voisins immédiats, c'est-à-dire à partir des voisins qui entrent dans son champ de communication sans fil. Comme le montre la figure 3.6, les voisins accessibles par le champ de communication sans fil du Nc, sont les nœuds N2 et N3. Donc, le prochain nœud de configuration sera parmi les nœuds N2 et N3.

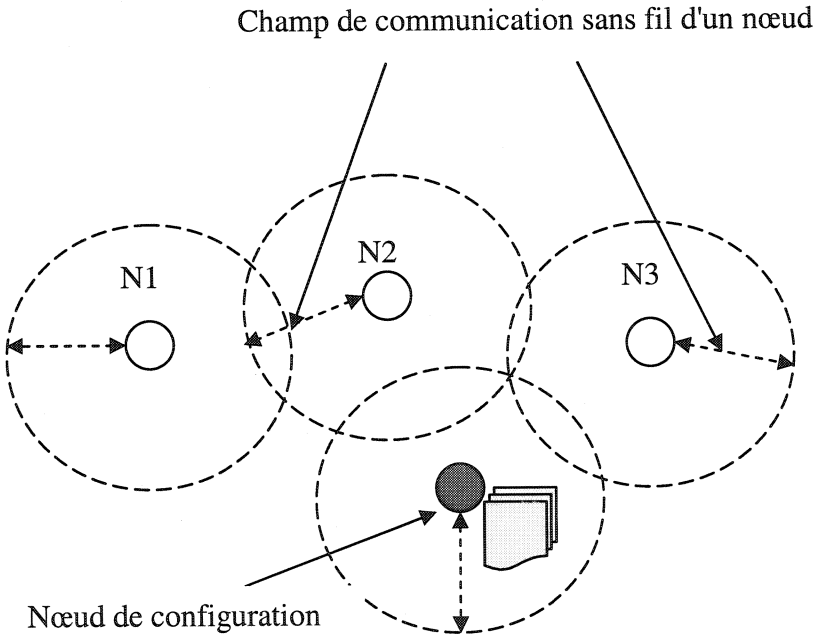


Figure 3.6 : Nœuds avec champ de communication sans fil

Plus précisément, le mécanisme de remplacement dans ce cas s'accomplit comme suit :

1- Lorsque le Nc décide de quitter, à son gré, le réseau, il choisit au hasard un de ses voisins immédiats comme successeur et l'informe de son départ en lui envoyant un message : Next_Nc.

2- Le nœud choisi comme successeur, envoie au Nc un Next_Nc_Reply pour l'informer qu'il a reçu son message Next_Nc.

3- Le Nc, envoie au successeur la table d'adresse TA.

4- Le nouveau Nc calcule un nouvel identificateur pour le réseau et diffuse un message New_Nc, pour informer le réseau qu'il est devenu le nouveau Nc.

L'organigramme de la figure 3.7 illustre graphiquement les étapes d'un départ intentionnel du nœud de configuration.

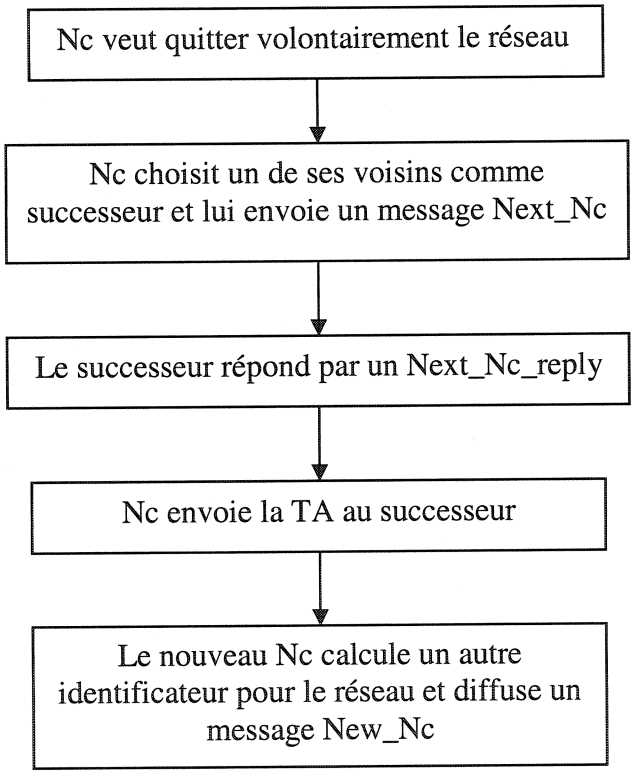


Figure 3.7 : Processus de départ volontaire du Nc

Nous signalons que le nœud contacté pour devenir Nc ne peut qu'accepter la demande car selon l'hypothèse 3, tous les nœuds du réseau sont en mesure de devenir des nœuds de configuration en cas de besoin.

- **Cas d'un départ inattendu**

Le nœud de configuration peut quitter le réseau mère sans préavis. Dans ce cas, le réseau continue son fonctionnement normal, et l'absence du nœud de configuration ne sera détectée que lorsqu'un service de configuration nécessite le Nc. Ceci est le cas par exemple lorsqu'un accompagnateur a besoin d'adresses de configuration. Un Nc est supposé en dehors du réseau si aucune réponse n'est reçue de sa part après un certain délai. À ce moment, le nœud qui a détecté cette absence devient candidat pour prendre la relève et déclenche le mécanisme de remplacement, qui se résume comme suit :

1- Détection de l'absence du Nc.

2- Diffusion de Update_Request : ce message informe le réseau qu'une mise à jour est en cours suite à un remplacement du Nc, ou tout simplement lorsqu'il y a manque d'adresses IP chez le Nc courant. Chaque nœud qui a reçu l'avis "Update_Request" doit répondre par un message : Update_Reply, qui contient des informations sur ses adresses de configuration.

3- Réception et analyse des réponses : les messages reçus vont servir à mettre à jour la table d'adresses TA. Et, si pendant cette étape aucun autre message relatif à un autre processus de remplacement de Nc n'est reçu, on passe à l'étape de mise à jour. Sinon, un des deux processus doit s'arrêter.

4- Mise à jour de la table d'adresses : en tenant compte du bloc d'adresses de base (hypothèse 1) et des réponses reçues.

5- Calcul d'un nouvel identificateur de réseau : Id_Network.

6- Diffusion du message New_Nc : la diffusion du message New_Nc informe le réseau qu'un nouveau nœud de configuration est devenu opérationnel. Ce message contient entre autres le nouvel identificateur de réseau.

Dans la figure 3.8, on donne une représentation schématique du processus de remplacement du Nc

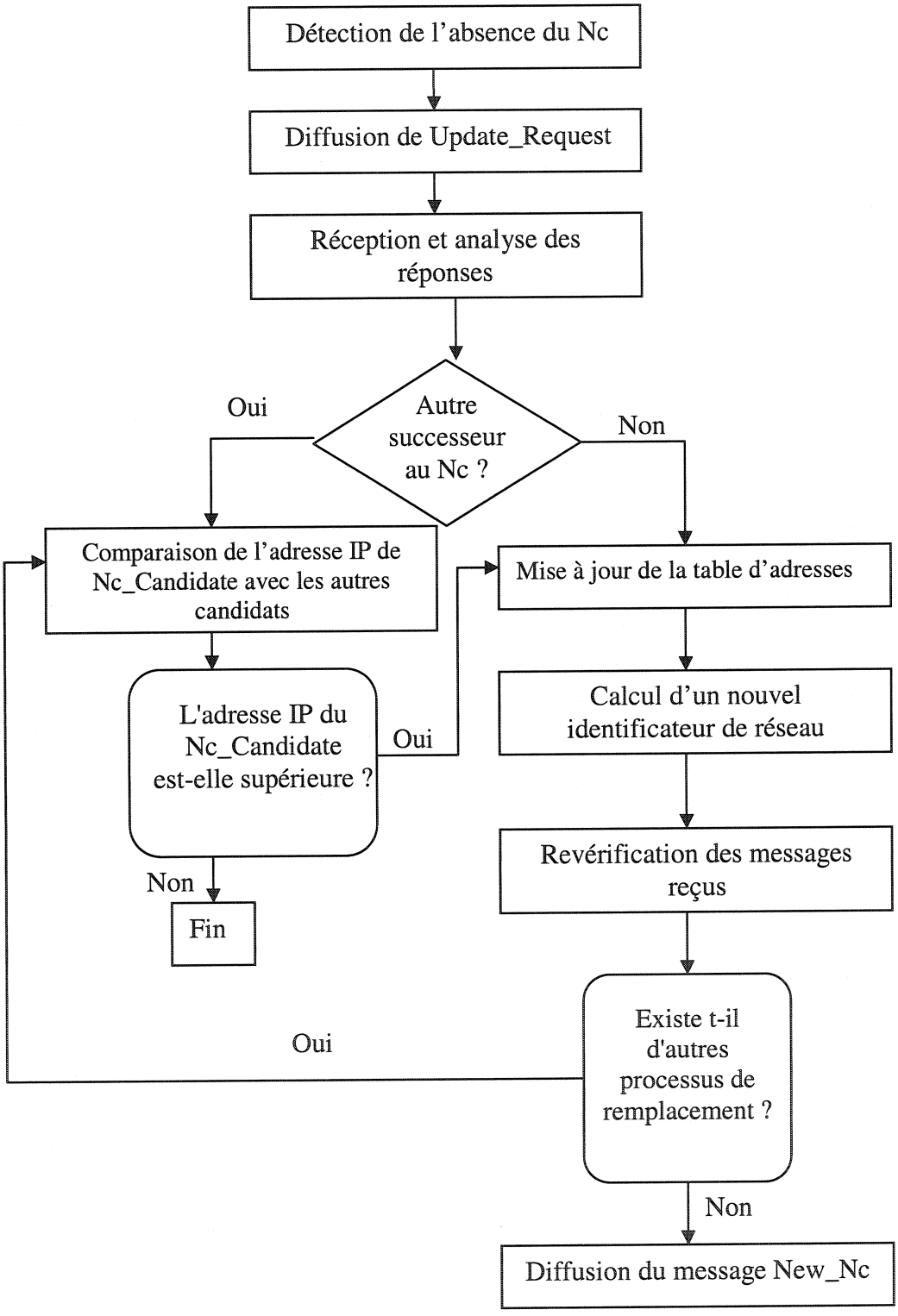


Figure 3.8 : Processus de remplacement d'un Nc

Le "Nc_Candidate" réfère au nœud du réseau qui a déclenché le mécanisme de remplacement. On rappel qu'à chaque fois qu'un changement de nœud de configuration survient, on doit calculer un nouvel identificateur de réseau.

3.5.4 Partitionnement du réseau ad hoc

On entend dire par partitionnement du réseau, le fait que ce dernier soit divisé en une ou plusieurs parties. Cette division peut survenir suite à des départs de nœuds, échelonné dans le temps, ou de façon instantanée suite à un accident par exemple. La figure 3.9, illustre un réseau mobile ad hoc en situation de partitionnement; d'un coté on trouve le réseau mère avec le Nc (réseau 1), et de l'autre coté le reste du réseau (réseau 2).

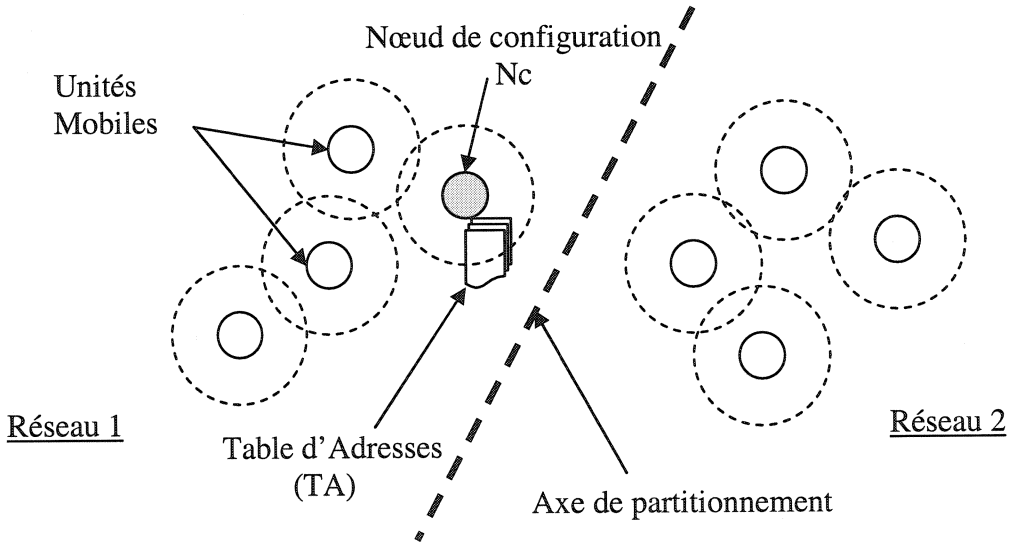


Figure 3.9 : Exemple de réseau en état de partitionnement

Les deux nouveaux réseaux vont continuer leur fonctionnement normal et la séparation ne va pas être détectée immédiatement. Afin de comprendre comment les deux réseaux vont réagir à ce partitionnement, nous allons étudier séparément deux cas de figure.

Cas du réseau 1

Dans ce cas le réseau va continuer son fonctionnement normal, la détection des nœuds partants ne se fera que pendant une phase de mise à jour de la table d'adresses du Nc (on parle ici d'une mise à jour par des messages Update_Reply). En effet, lorsque le nœud de configuration procède à une mise à jour de la table TA (suite à un remplacement du Nc par exemple), les nœuds du réseau 2 ne seront plus en mesure de

répondre à la demande de mise à jour du Nc, vu qu'ils sont suffisamment éloignés du réseau mère. Par conséquent, ils seront ignorés par le réseau 1 et leurs adresses IP seront ainsi récupérées.

Cas du réseau 2

Ce réseau va continuer lui aussi à fonctionner même s'il n'a plus de nœud de configuration. Lorsque le Nc est sollicité pour un service de configuration, le réseau se rendra compte de l'absence de son nœud de configuration. Dans ce cas, le mécanisme de remplacement sera déclenché (départ involontaire du Nc). Ainsi, comme nous l'avons mentionné précédemment, ledit mécanisme entame une procédure de mise à jour des adresses IP. Pendant cette mise à jour, le départ des autres nœuds sera détecté et leurs adresses seront récupérées. La figure 3.10 montre les étapes entreprises dans ce cas ci.

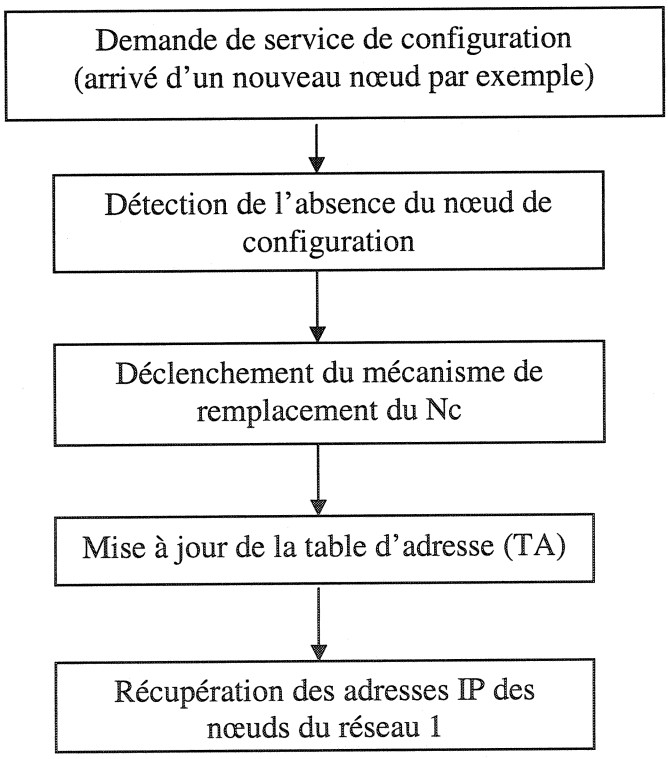


Figure 3.10 : Réaction d'un réseau sans Nc suite à un partitionnement

3.5.5 Fusion

On parle de fusion lorsqu'un ou plusieurs réseaux se rapprochent de façon à former une intersection non vide entre eux. Ce genre de scénario est illustré par la figure 3.11. Notre but dans cette partie est de proposer des solutions pour que les unités mobiles des réseaux en fusion puissent communiquer entre elles sans qu'il y ait risque de conflits d'adresses. En réalité, lors d'une fusion, on est confronté à l'un des cas suivants :

a) Fusion avec deux nœuds de configuration

Dans ce cas, chacun des réseaux en fusion possède un nœud de configuration comme le montre la figure 3.11.

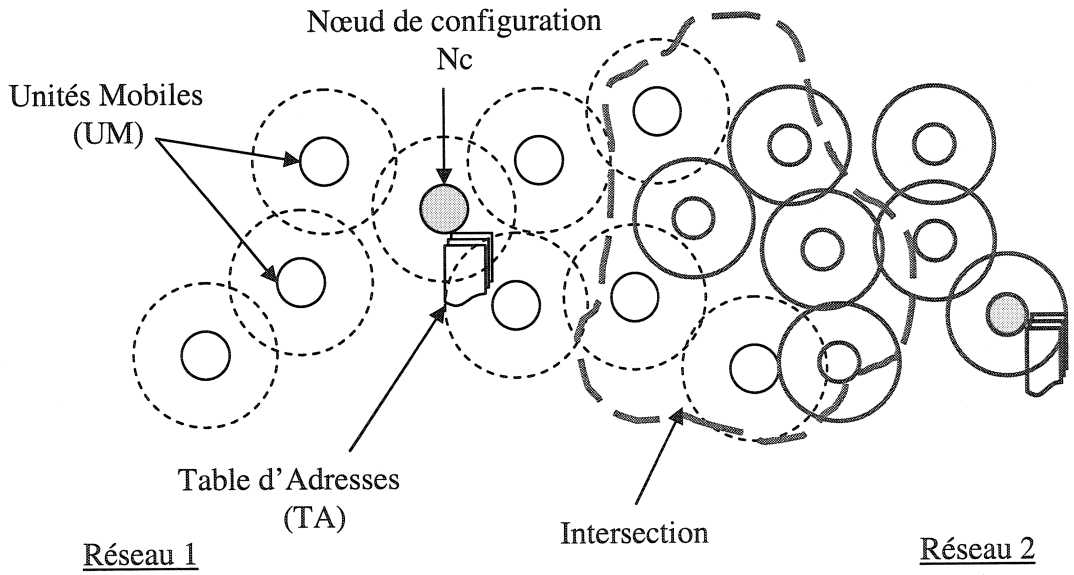


Figure 3.11 : Exemple de réseaux en fusion avec deux nœuds de configuration

D'habitude, une fusion est détectée lorsque deux nœuds appartenant à des réseaux différents entrent en communication à travers leurs interfaces sans fil respectives. Dès que ce genre de situation survient, les nœuds qui ont détecté la fusion s'échangent des messages de "Hello" pour s'informer, de part et d'autre, de l'état des réseaux en cours de fusion. On entend dire par état du réseau le fait que ce dernier possède ou non un nœud

de configuration. Afin de résoudre le problème de fusion dans ce cas, on propose le mécanisme suivant:

1- Détection de la fusion et détermination de l'état du réseau.
2- Les nœuds de configurations de chaque réseau sont informés qu'une fusion est en cours.
3- Les Nc comparent mutuellement leurs identificateurs de réseaux.
4- Le nœud de configuration du réseau ayant le plus grand identificateur maintient son état (garde le même identificateur de réseau) et récupère la table d'adresses de l'autre réseau.
5- Détection des conflits d'adresses et proposition de nouvelles adresses pour les nœuds en conflits. On s'assure que les nœuds en conflits d'adresses se désengagent de toutes communications avant de changer leurs adresses.
6- Diffusion du message New_Nc sur le nouveau réseau.

Dans le processus de la figure 3.12, on résume le mécanisme susmentionné.

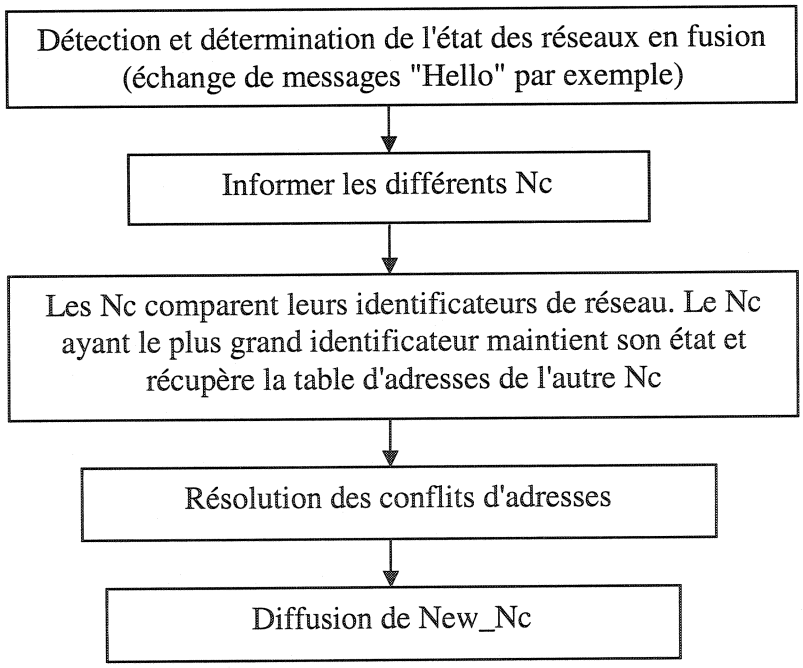


Figure 3.12 : Mécanisme de fusion avec deux nœuds de configuration

b) Fusion avec un seul nœud de configuration

Les réseaux qui fusionnent dans ce cas correspondent à un réseau avec nœud de configuration et un autre réseau sans Nc comme l'indique la figure 3.13.

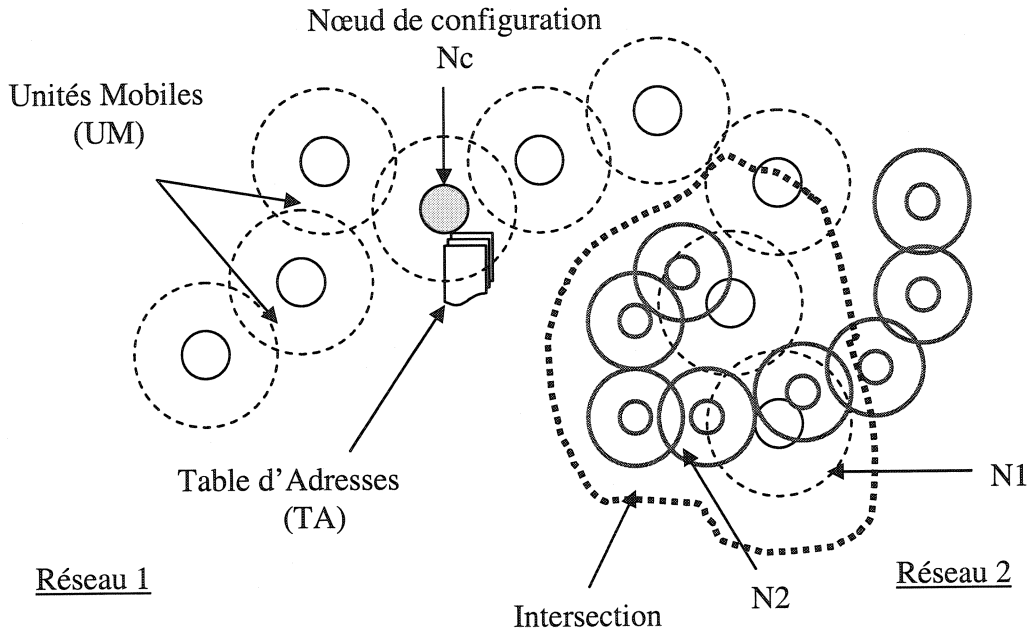


Figure 3.13 : Exemple de réseaux en fusion avec un seul Nc

La figure ci-dessus montre d'une part un réseau avec nœud de configuration (Réseau 1) et d'autre part un réseau sans nœud de configuration (Réseau 2). L'intersection non vide qui s'est formée entre le réseau 1 et le réseau 2 témoigne d'un phénomène de fusion. Encore une fois, la détection de la fusion se fera par les nœuds de l'intersection comme dans le cas précédent.

Pour résoudre le problème de fusion dans ce cas, on propose le processus suivant :

1- Détection de l'état de la fusion par échange de messages de "Hello" au niveau de l'intersection des deux réseaux.

2- Le Nc du réseau 1 est informé de l'état de la fusion.

3- Le Nc récupère les adresses du réseau 2, à travers les nœuds de l'intersection comme l'explique l'exemple suivant : soit N1 et N2 deux nœuds appartenant respectivement au réseau 1 et réseau 2. Le Nc choisit N1 comme intermédiaire;

N1 envoie une demande de récupération d'adresses à N2, qui diffuse un message Update_Request dans son réseau mère; Et, après avoir reçu les réponses, N2 les envoie au noeud N1 qui les achemine à son tour au Nc.

4- Les adresses IP des nœuds du réseau 1 en conflit sont changées. On note que les changements d'adresses ne devront être entamés que lorsque les nœuds concernés se libèrent de leurs communications courantes.
5- Diffusion d'un message New_Nc.

Dans la figure 3.14, on donne une illustration graphique du processus cité plus haut.

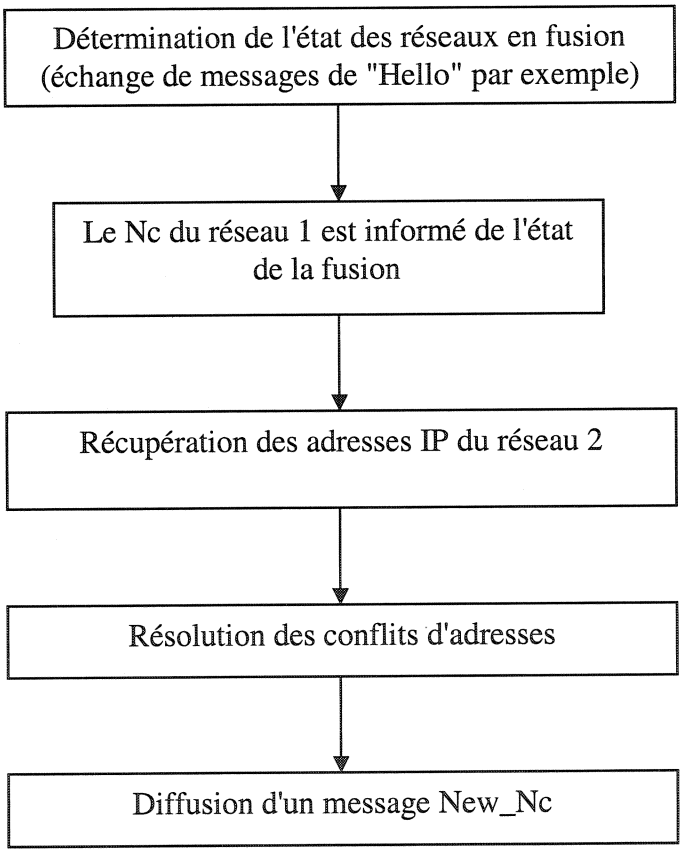


Figure 3.14 : Processus de fusion avec un seul nœud de configuration

c) Fusion sans nœud de configuration

Dans ce cas les réseaux en fusion sont dépourvus de nœuds de configuration comme le montre la figure 3.15. Dans cette figure, on remarque que ni le réseau 1, ni le

réseau 2 ne disposent de nœud de configuration, et en plus une intersection non vide s'est constituée entre eux. Pour pouvoir résoudre le problème de fusion dans une telle situation, on propose le mécanisme suivant :

1- Détection de l'état de fusion : dans ce cas les nœuds de l'intersection vont essayer de part et d'autres d'informer leurs Nc respectives de l'existence d'une fusion. Évidement, les tentatives de contacter les Nc vont échouer du fait qu'aucun nœud de configuration n'existe sur les deux réseaux.
2- Les deux nœuds qui ont découvert la fusion (nœud 1 et le nœud 2 par exemple) vont déclencher de part et d'autre le mécanisme de remplacement du Nc.
3- On retombe sur le cas d'une fusion avec deux nœuds de configuration et on applique à ce moment le mécanisme correspondant.

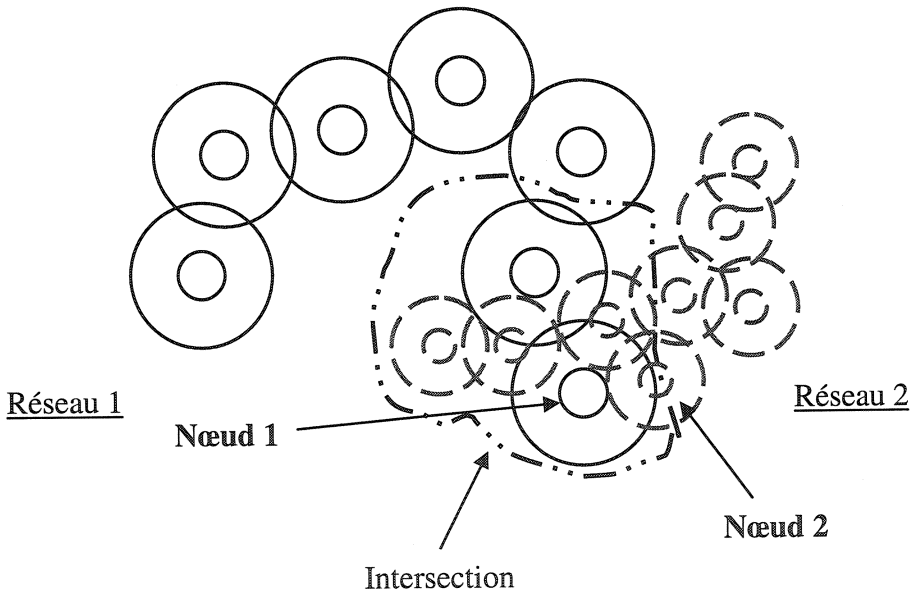


Figure 3.15 : Exemple de réseaux en fusion sans Nc

D'après les différents cas de fusion que nous avons traité durant cette partie, on peut dire que la fusion n'est pas un problème facile, et nécessite plusieurs mécanismes pour le résoudre. Cependant, dans notre approche de configuration, c'est la configuration qui est prioritaire. Ceci veut dire que si on a des demandes de

configurations à satisfaire et un phénomène de fusion à résoudre, nous allons assurer le service de configuration en premier et ensuite traiter les éventuelles fusions. Dans la figure 3.16, on propose une récapitulation des différents problèmes liés au caractère dynamique des réseaux mobiles ad hoc.

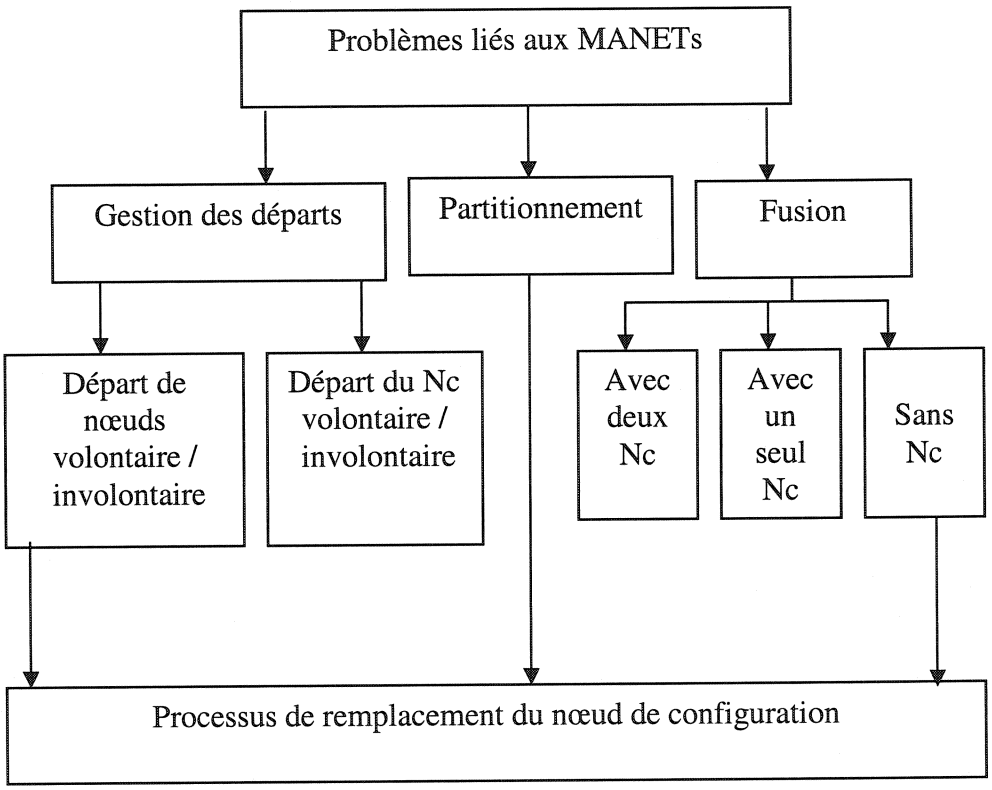


Figure 3.16 : Récapitulation des problèmes liés aux MANETs

Ces problèmes concernaient particulièrement, la gestion des départs, le partitionnement et la fusion. On signale que la récupération des adresses se fait implicitement lors des mises à jour des tables d'adresses. Dans le chapitre suivant, nous allons juger de la qualité de notre approche de configuration moyennant des simulations et des analyses de performances.

Chapitre 4
Implémentation et analyse de performances

Durant le précèdent chapitre, nous avons exposé avec plus de détails les différentes étapes de notre protocole de configuration pour MANETs : APM. L'idée de ce protocole était construite autour d'une allocation directe des adresses IP (l'avis des autres composantes du réseau n'est pas sollicité), et sur une gestion centralisée des problèmes liées à la nature dynamique des MANETs. Les problèmes en question concernaient les départs des nœuds mobiles, la récupération des adresses IP non utilisées dans le réseau, le partitionnement et la fusion. Dans le présent chapitre, nous allons discuter de la pertinence de notre solution vis à vis des méthodes de configuration basées sur la diffusion ou sur ce qu'on appelle communément : le vote. Plus exactement, nous implémenterons dans un premier temps notre protocole APM, ensuite nous ferons de même pour celui de [Nesargi 2002]. Le choix de cette méthode comme support de comparaison se justifie par le fait qu'elle est la plus récente et la plus complète parmi les méthodes proposées dans la catégorie des méthodes de configuration avec détection de conflits. En effet, en plus de proposer un mécanisme d'allocation d'adresses, elle traite les problèmes de partitionnement et de fusion qui ne sont pas traités nulle part auparavant dans cette catégorie de méthodes. Les implémentations seront faites moyennant le simulateur GLOMOSIM qui est conçu spécialement pour les réseaux mobiles ad hoc. Dans le reste de ce chapitre, nous traiterons les points suivants : nous allons décrire dans un premier temps, l'environnement d'implémentation; ensuite, nous parlerons de la mise en œuvre de notre protocole et de celui de Nesargi sous GLOMOSIM. Enfin, nous présenterons les plans d'expériences, les résultats et les interprétations.

4.1 Description de l'environnement d'implémentation

Une fois la solution théorique mise en place, on doit la valider sur un cas réel pour juger de son efficacité. Malheureusement, dans le domaine des réseaux, il n'est pas toujours possible de faire des validations sur des cas réels car celles-ci nécessitent en général des infrastructures inexistantes ou des coûts très élevés. C'est la raison pour laquelle on se contente de simulations à l'aide d'outils conçus spécialement à cet effet. Dans notre cas, nous utiliserons GLOMOSIM pour analyser les performances de notre proposition de configuration. Le choix de GLOMOSIM se justifie par le fait qu'il est tout d'abord reconnu dans les milieux de recherche. En plus, il est en grande partie dédié aux réseaux mobiles ad hoc. Et, il est en plus, gratuit et open source, c'est à dire qu'on peut modifier son code pour l'adapter à nos propres besoins.

4.1.1 Bref aperçu sur GLOMOSIM

GLOMOSIM (Global Mobile Information System Simulator) est un environnement de simulation pour les réseaux sans fils [Glomosim 2001]. Il est basé sur l'utilisation du parallélisme discret d'événements pour optimiser le temps de simulation.

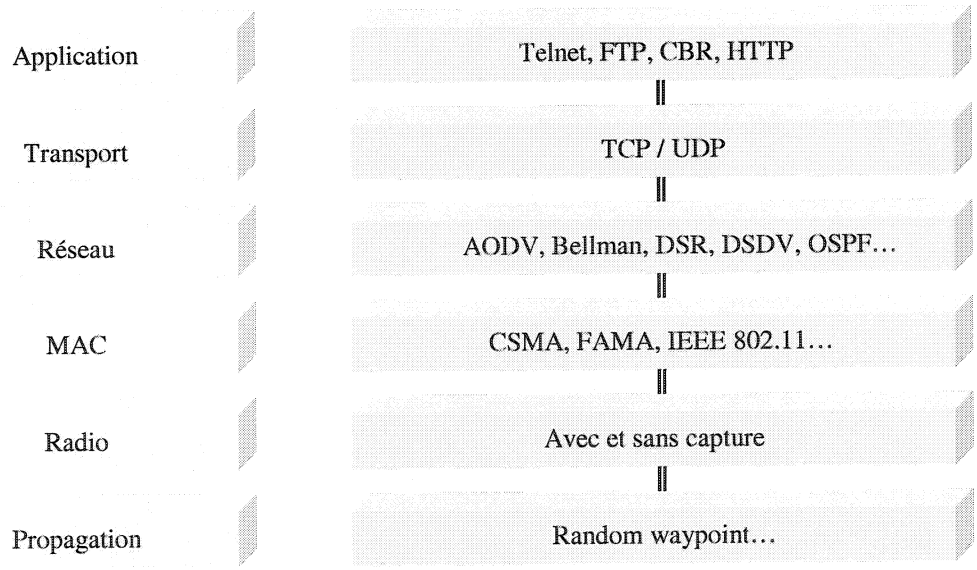


Figure 4.1 : Couches de simulation de GLOMOSIM

Cette façon de faire, fait de GLOMOSIM un simulateur évolutif dans le sens où chaque utilisateur peut intégrer ses propres événements au système de base. Il supporte aussi la mobilité, ce qui permet à chaque utilisateur de définir sa propre stratégie de mobilité. Pour répondre à divers besoins de simulation et pour en faciliter l'implémentation, GLOMOSIM utilise une approche de couche [Martin 1999] comme le montre la figure 4.1.

Afin d'assurer le parallélisme, GLOMOSIM se base sur le langage PARSEC (Parallel Simulation Environment for Complex Systems) qui est lui même basé sur le langage C. Le PARSEC offre un environnement de programmation semblable à celui de C, des descriptions plus exhaustives sont disponibles dans [Parsec 1998].

4.1.2 Fonctionnement de GLOMOSIM

Dans cette partie nous allons nous contenter de montrer le fonctionnement général de GLOMOSIM sans entrer dans les détails de son implantation. La figure 4.2 montre de façon schématique ce fonctionnement.

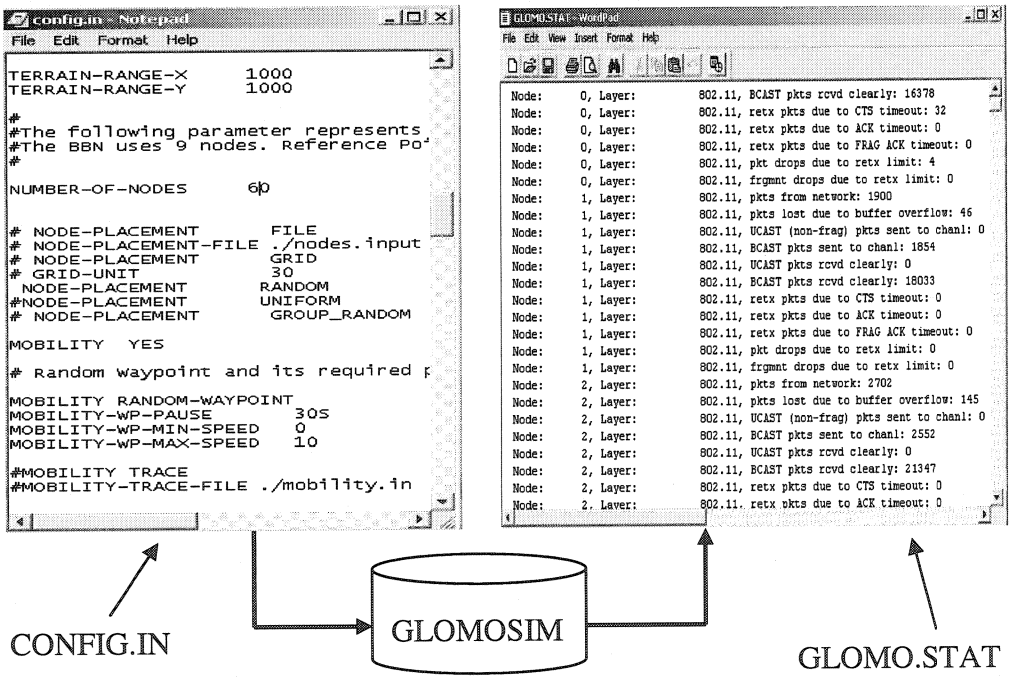


Figure 4.2 : Fonctionnement général de GLOMOSIM

Pour exécuter une simulation, GLOMOSIM lit le fichier de configuration : *CONFIG.IN*, puis, il accomplit les traitements nécessaires et génère un fichier résultats appelé *GLOMO.STAT*.

Le *CONFIG.IN* contient les différents paramètres de configuration nécessaires pour une simulation. Parmi ces paramètres on trouve par exemple :

- *SIMULATION TIME* : durée de la simulation,
- *PARTITION-NUM* : nombre de partition du réseau,
- *TERRAIN-RANGE-X* : longueur de l'espace de simulation,
- *TERRAIN-RANGE-Y* : Largueur de l'espace de simulation,
- *NUMBER-OF-NODES* : Nombre de nœuds

Ceci n'énumère pas tous les paramètres de configuration, il en existe d'autres concernant notamment les stratégies de mobilités les statistiques etc. L'utilisateur peut ajouter lui même ses propres paramètres personnifiés. Quant au fichier *GLOMO.STAT*, il est généré à la fin d'une simulation et il contient les différentes statistiques préalablement spécifiées dans le fichier d'entré : *CONFIG.IN*.

Le GLOMOSIM est constitué en réalité d'une collection de modules comme le montre la figure 4.3.

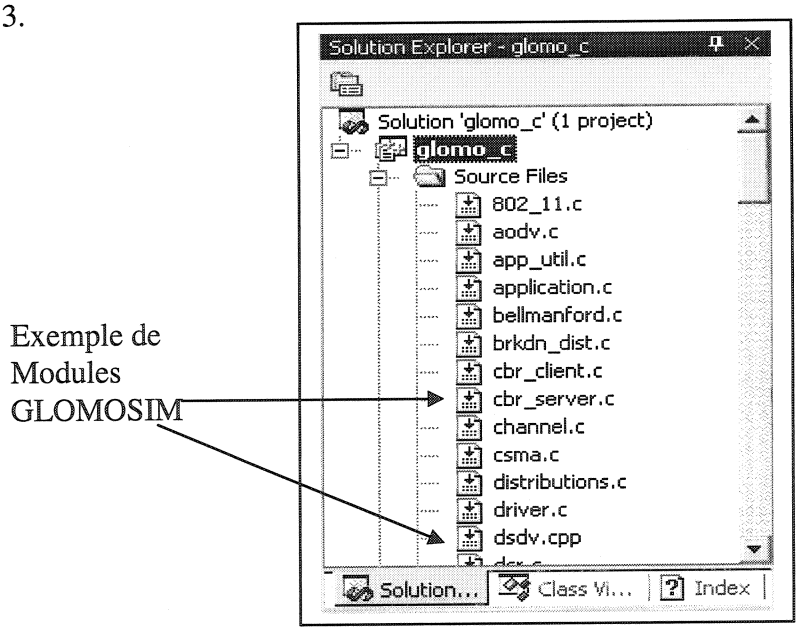


Figure 4.3 : Exemple de Modules GLOMOSIM

Chaque module est dédié à une fonction donnée (protocoles de routage, méthodes d'accès au réseau, couches, etc.). En résumé, GLOMOSIM se base dans son fonctionnement sur les points suivants :

- Entity driver () : Cette entité, qui est l'équivalent du main () en C, se trouve le fichier *driver.c*. A partir de cette entité, on peut créer d'autres entités, et construire ainsi l'environnement de simulation.
- Création et initialisation des nœuds : la structure nœud est déjà prédéfinie (Glomonode), et c'est l'élément clé de la simulation.
- Initialisation des couches GLOMOSIM.
- Messages ou événements : ils représentent les échanges qui peuvent survenir à la fois entre des entités, des nœuds et éventuellement des couches. L'utilisateur a la possibilité d'ajouter ses propres messages et événements.
- Collection des statistiques et terminaison de la simulation.

4.2 Implémentation et mise en œuvre

Dans cette partie, nous allons décrire l'implémentation du protocole que nous avons proposé. Ensuite nous ferons de même pour le mécanisme de configuration de [Nesargi 2002]. Comme nous l'avons mentionné auparavant, le choix de la méthode de Nesargi comme support de comparaison, revient au fait qu'elle soit la plus complète de point de vue traitement des problèmes liés à la nature dynamique des MANETs.

4.2.1 Métriques

En parallèle de l'implémentation des processus relatifs à la configuration, nous allons aussi recueillir des informations concernant la simulation en spécifiant certains critères de performance. Les critères en question sont :

a) La latence

La latence fait référence au temps nécessaire pour qu'une demande de configuration soit satisfaite. Ce paramètre est fort intéressant, car plus le temps d'attente est faible, meilleur sera le protocole de configuration. Evidemment, ce critère fait une évaluation à base de temps, et la qualité d'un mécanisme de configuration dépend aussi

de sa consistance et de son aptitude à traiter les problèmes liés au caractère aléatoire des MANETs.

b) Nombre de messages

Le nombre de messages échangés entre nœuds a un impact direct sur la surcharge du réseau. Donc, ce paramètre va nous permettre de quantifier l'influence d'un protocole de configuration sur le trafic du réseau.

4.2.2 Mise en œuvre du protocole APM

Dans la figure 4.4, on résume le mécanisme de configuration que nous avons présenté au chapitre 3. On retrouve d'une part les différents processus qui constituent ledit mécanisme et d'autre part les principales interactions qu'il subit.

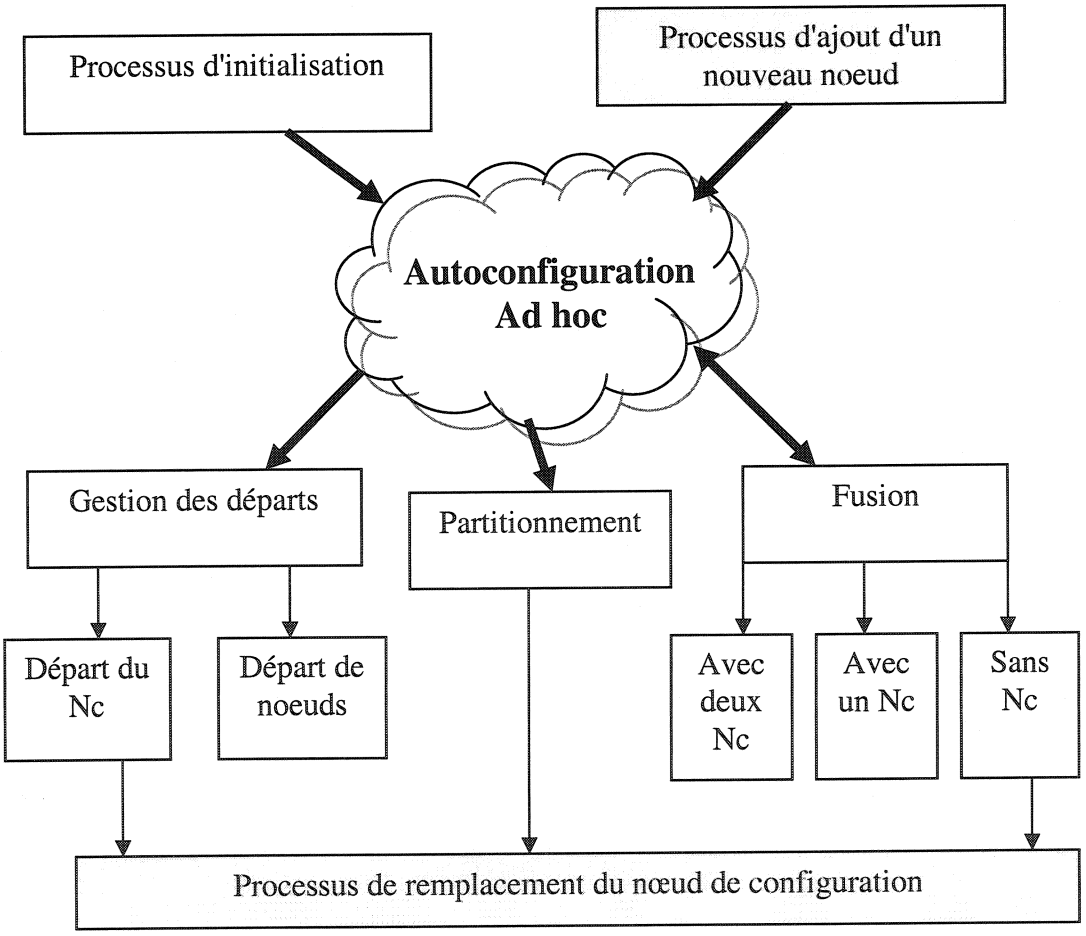


Figure 4.4 : Principaux processus du protocole APM

Pendant la survie d'un réseau mobile ad hoc, le processus d'initialisation ne s'exécute qu'une seule fois. Donc il ne présente pas un intérêt particulier dans l'analyse de performance. En revanche le processus d'ajout de nouveaux nœuds sera sollicité continuellement sur le réseau. Il jouera donc un rôle important dans les prochaines analyses de performances. Tandis que les autres processus à savoir: la gestion des départs, le partitionnement et la fusion interagissent de façon permanente avec le processus de remplacement du nœud de configuration. Ceci montre que ce processus sera lui aussi un paramètre clé dans les évaluations futures. Dans ce qui suit, notre intérêt d'implémentation sera particulièrement orienté vers le processus d'ajout de nouveaux noeuds et, vers celui de remplacement du Nc.

Une implémentation sous GLOMOSIM nécessite une compréhension approfondit des différents modules qui le constitue ainsi que des interactions qui existent entre eux.

Dans notre cas, nous allons intervenir essentiellement au niveau des modules *glomo.c*, *drivers.c*, *802_11.c*, *node.c*, *radio.c* et *message.c*.

a) Implémentation du processus d'ajout d'un nouveau nœud

L'implémentation de ce processus consiste à ajouter certaines fonctions aux modules de GLOMOSIM et d'en modifier d'autres. À titre d'exemple, on décrit ci-après quelques unes des méthodes manipulées :

- L'ajout de noeuds à un réseau en cours d'utilisation, *add_new_node()*;
- Manipulation des données d'un nouveau nœud, *GLOMO_GetNodeData ()* ;
- Initialisation d'un nouveau nœud, *initiate_node()* ;
- Initialisation des couches : *GLOMO_PropagateInit()*, *GLOMO_ChannelInit()*, *GLOMO_RadioInit()*, *GLOMO_MacInit()*, *GLOMO_NetworkInit()*, *GLOMO_TransportInit()*, *GLOMO_AppInit()*.
- Mobilité : *GLOMO_MobilityInit()*.
- Synchronisation de la simulation : *GLOMO_TimerInitialize()*,
- Manipulation de messages: *GLOMO_MsgAlloc()*, *GLOMO_MsgInfoAlloc()*, *GLOMO_MsgGetLayer()*.
- Envoi de messages aux voisins, *GLOMO_MsgSend_message()*;

- Récupération d'événements ou de messages, *read_message()*;
- Calcul du délai d'un lien, *link_delay()*;
- Capture des messages générés, *message_capture()*;
- Détection de l'état du réseau, *status_detection()* ;
- Edition de statistiques, GLOMO_PrintStat() etc…

En plus de l'implémentation du processus d'ajout d'un nouveau nœud, nous avons aussi ajouté des mesures de latence et de messages échangés suite à un ajout.

- **Latence d'ajout d'un nouveau noeud**

  Une fois qu'un nouveau noeud envoie une requête de configuration à ses voisins, le temps nécessaire pour recevoir une réponse de configuration dépend de la disponibilité d'adresses libres chez l'accompagnateur.

  - Cas où l'accompagnateur possède des adresses libres

  La latence dans ce cas est donnée par :

  $T_{latence} = T_1 + T_2 + T_3 + T_4 + T_{trait\_acc}$ (1)

  La figure 4.5 montre les échanges effectués entre le nouveau nœud et son accompagnateur.

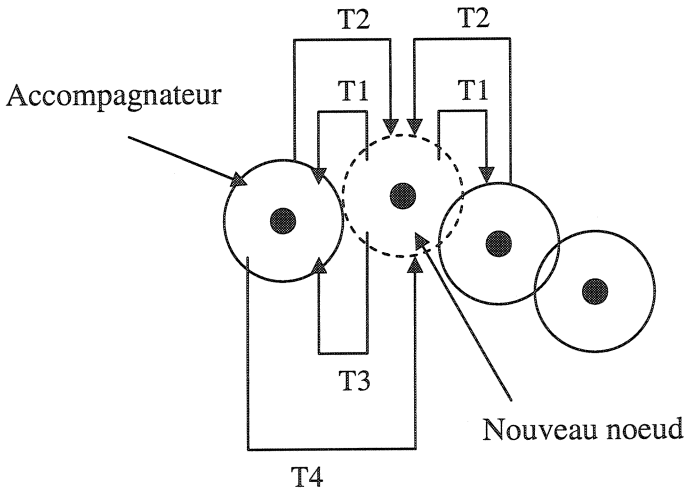


Figure 4.5 : Les différents paramètres temps d'une configuration directe

Où :

T_1 : Temps pour qu'une demande de configuration atteigne les voisins;

T_2 : Temps de réponse des voisins;

T_3 : Temps de confirmation du choix d'un accompagnateur;

T_4 : Temps d'acheminement des informations de configurations au nouveau nœud;

T_{trait_acc} : Temps de traitement au niveau de l'accompagnateur.

- **Cas où l'accompagnateur ne possède pas d'adresses libres**

Dans ce cas la latence est donnée par :

$$T_{latence} = T_1 + T_2 + T_3 + T_4 + T_{trait\_acc} + T_{trait\_Nc} + T_{req} + T_{rep} \quad (2)$$

Avec :

T_1 : Temps pour qu'une demande de configuration atteigne les voisins;

T_2 : Temps de réponse des voisins;

T_3 : Temps de confirmation du choix d'un accompagnateur;

T_4 : Temps d'acheminement des informations de configurations au nouveau nœud;

T_{req} : Temps d'acheminement d'une requête de configuration de l'accompagnateur au nœud de configuration;

T_{rep} : Temps d'acheminement d'une réponse du nœud de configuration à l'accompagnateur;

T_{trait_acc} : Temps de traitement au niveau de l'accompagnateur;

T_{trait_Nc} : Temps de traitement au niveau du Nc.

Les différents paramètres de temps cités plus haut sont illustrés dans la figure 4.6.

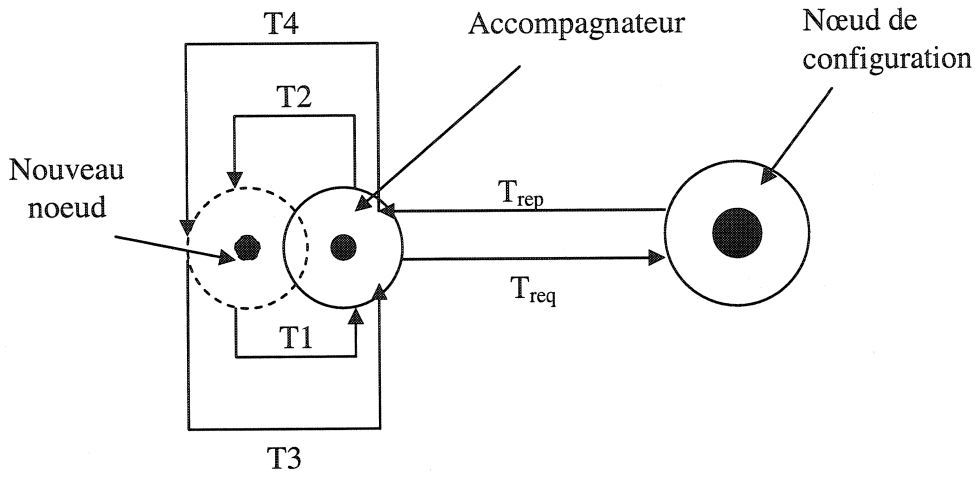


Figure 4.6 : Les différents paramètres temps dans le cas où l'accompagnateur ne possède pas d'adresses libres

- **Messages générés par l'ajout d'un nouveau nœud**

Les messages échangés dans ce processus concernent essentiellement l'accompagnateur et le nœud de configuration. En effet, les échanges entre le nouveau nœud et son accompagnateur se font de façon directe et locale, dans le sens où ils ne passent pas par des nœuds intermédiaires. Par contre, les positions du Nc et de l'accompagnateur changent de manière continue, donc les messages sont supposés transiter à travers plusieurs nœuds intermédiaires. Dans la pratique, on utilise la fonction *GLOMO_MsgAlloc()* pour simuler la génération de nouveaux messages destinés à la couche MAC et la fonction *GLOMO_MsgSend()* pour simuler les envois. Le traitement et le dénombrement des messages sont assurés par les fonctions *Mac802_11BroadcastTransmitted()* et *Mac802_11ProcessFrame()*.

b) Implémentation du processus de remplacement du Nc

Le processus de remplacement du Nc est déclenché, par exemple, lorsque le nœud accompagnateur n'arrive pas à contacter le Nc. Il peut être aussi amorcé lors de la détection d'un phénomène de fusion sans Nc. Ce dernier cas a été préalablement discuté

dans le chapitre 3. Dans ce qui suit, nous allons voir en quoi consiste la latence et les messages échangés dans ce cas.

- **Latence du processus de remplacement du Nc.**

  Les différents paramètres temps de la formule de latence sont donnés dans la figure 4.7. Le délai de remplacement du Nc est donné par :

  $TNc_{latence} = T_1 + T_2 + T_3 + T_4 + T_{req} + T_{timer} + T_{trait\_acc} + T_{bcast} + T_{rep}$ (3)

  T_1 : Temps pour qu'une demande de configuration atteigne les voisins;

  T_2 : Temps de réponse des voisins;

  T_3 : Temps de confirmation du choix d'un accompagnateur;

  T_4 : Temps d'acheminement des informations de configurations au nouveau nœud;

  T_{req} : Temps d'acheminement d'une requête de configuration au Nc;

  T_{trait_acc} : Temps du traitement de l'accompagnateur ou du nœud qui détecté l'absence du Nc;

  T_{bcast} : Temps pour faire une diffusion sur le réseau;

  T_{rep} : Temps de réception des réponses du réseau;

  T_{timer} : Temps d'attente avant de déclencher le mécanisme de remplacement du Nc.

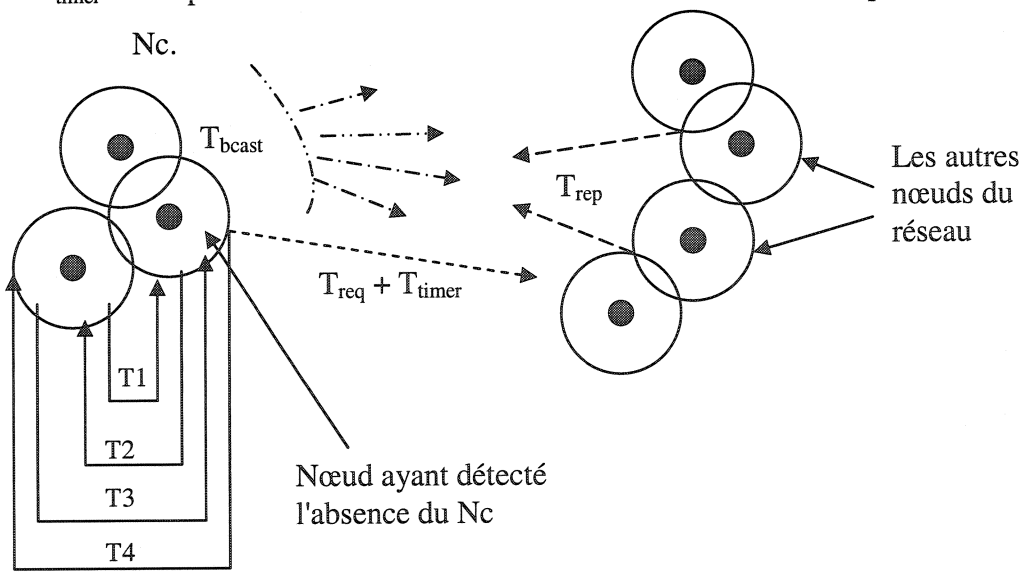


Figure 4.7 : Les différents paramètres temps du processus de remplacement du Nc

- **Messages générés par le processus de remplacement du Nc**

  Dans ce cas, deux principaux échanges de messages surviennent. Tout d'abord lors de la diffusion du message : "Update_Request" par le nœud ayant détecté l'absence du Nc; ensuite, pendant la diffusion du message : "New_Nc", par le nouveau Nc, pour informer le reste du réseau qu'un autre Nc à entrer en fonction. Comme dans le cas précédent, les principales fonctions utilisées pour la simulation sont: *GLOMO_MsgAlloc ()*, *GLOMO_MsgSend ()*, *Mac802_11BroadcastTransmitted ()* et *Mac802_11ProcessFrame ()*.

c) Implémentation de la méthode de Nesargi

Dans ce cas aussi, nous allons tenir compte du temps nécessaire pour attribuer une adresse IP à une nouvelle unité mobile et du trafic généré par cette allocation en terme de messages échangés. Les différents paramètres temps de cette méthode sont présentés dans la figure 4.8.

- **Latence de la méthode de Nesargi**

  La latence est donnée dans ce cas par la formule :

  $T_{latence} = T_1 + T_2 + T_3 + T_4 + T_{trai\_acc} + T_{bcast} + T_{rep}$ (4)

  Avec :

  T_1 : Temps d'une demande de configuration destinée au voisin;

  T_2 : Temps de réponse des voisins;

  T_3 : Temps de confirmation et du choix d'un accompagnateur;

  T_4: Temps d'acheminement de l'adresse IP par l'accompagnateur;

  T_{trait_acc} : Temps du traitement de l'accompagnateur;

  T_{bcast} : Temps pour faire une diffusion sur le réseau;

  T_{rep} : Temps de réception des réponses du réseau.

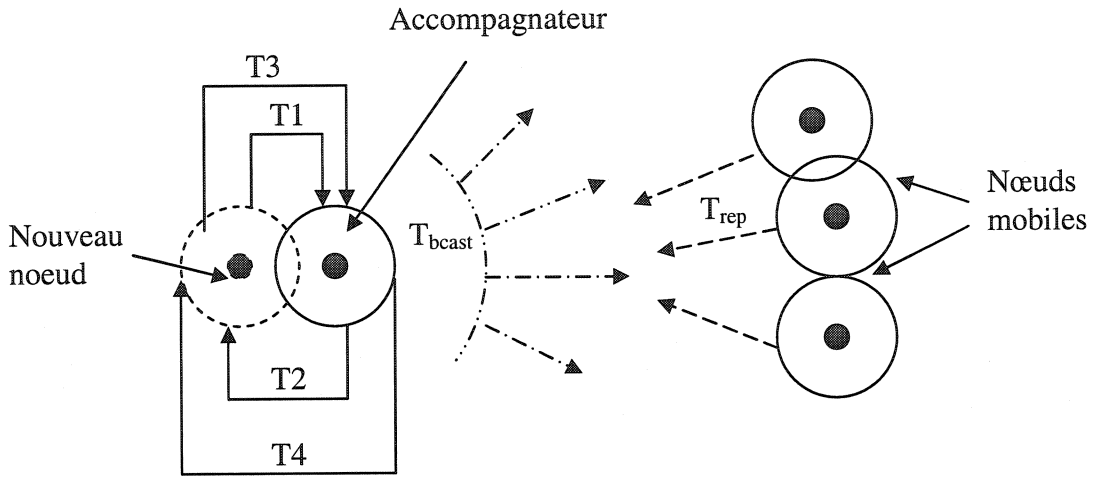


Figure 4.8 : Les différents paramètres temps de la méthode de Nesargi

- **Messages générés par la méthode de Nesargi**

  Le trafic généré par cette méthode de configuration est dû principalement à la diffusion de messages effectuée pour valider une adresse IP, et pour informer le reste du réseau qu'une adresse IP a été allouée. Ajoutons à ceci, les messages de confirmation envoyés par le reste des composants du MANET.

Après avoir présenté les grandes lignes de l'implémentation, sous GLOMOSIM, des principaux processus du protocole APM et de celui de Nesargi, nous allons présenter dans ce qui suit les plans d'expériences avec lesquels nous avons travaillé, ainsi que les résultats obtenus.

4.3 Expériences de simulation, résultats et interprétations

Dans cette partie, nous allons décrire les expériences de simulation à réaliser; ensuite, pour chaque expérience, nous allons présenter et discuter les résultats obtenus. Les expériences de simulation seront axées essentiellement sur la latence et sur le trafic généré en terme de messages. Pour accomplir nos expériences de simulation, nous disposons des supports suivants:

- Logiciels
    - GLOMOSIM version 2.0
    - PARSEC version 1.1
    - Microsoft Development Environment 2002 version 7
- Matériel
    - Ordinateur menu de processeur Pentium 4
    - 1.8 Mhz, 128 Mo RAM, 40 Go de disque, 512 ko de caches
- Paramétres de configuration
    - Terrain de simulation : la dimension de l'espace de simulation est de 1000 mètres de large, par 1000 mètres de long.
    - Durée de simulation : 2 heures.
    - Portée du signal : 250 mètres.
    - Mode de déplacement : aléatoire.
    - Stratégie de mobilité : le nœud mobile peut se déplacer avec des vitesses allant de 0 à 30 mètres par seconde dans toutes les directions, et une pause de 5 secondes.
    - Modèle de propagation : Free-Space, c'est à dire on utilise un terrain sans éléments d'obstruction.

4.3.1 Expériences relatives à la latence

Le but de ces expériences est de mesurer le temps nécessaire pour qu'une demande de configuration soit satisfaite, en fonction de la taille du réseau. Dans un premier temps, nous allons mesurer les latences du protocole APM, ensuite nous ferons de même pour celui de Nesargi.

a) Cas du protocole APM

Nous avons vu dans le chapitre 3 que, le protocole APM, effectue une approche de configuration hiérarchique, dans le sens où un nouveau nœud peut être configuré par son accompagnateur de l'une des façons suivantes :

1- Sans consultation du Nc : ceci est le cas lorsque l'accompagnateur possède suffisamment d'adresses libres.

2- Avec consultation du Nc, mais sans attente : c'est le cas lorsque l'accompagnateur ne dispose pas d'assez d'adresses libres.

3- Avec consultation du Nc: cas où l'accompagnateur ne dispose d'aucune adresse IP.

Du point de vue délai d'attente, le cas 1 et 2 ont la même latence, car dans les deux cas, la configuration se fait de façon directe entre le nouveau et son accompagnateur. Cependant, dans le cas 3, un délai supplémentaire est généré suite à la consultation du Nc. Dans ce qui suit, nous allons expérimenter ces deux cas de figure (cas où l'accompagnateur possède au moins une adresse libre, et le cas où il possède aucune adresse libre).

- **<u>Expérience 1</u> : cas où l'accompagnateur possède au moins une adresse libre.**

Dans ce cas la configuration va se faire de manière directe entre le nouveau nœud et son accompagnateur. Dans la section précédente, nous avons vu que la latence est donnée par :

$$T_{latence} = T_1 + T_2 + T_3 + T_4 + T_{trait\_acc}$$

L'influence du temps de traitement de l'accompagnateur (T_{trait_acc}) est négligeable à cause du fait que les composants du réseau sont dotés de bonnes capacités de traitements (hypothèse 3). La figure 4.9 montre les résultats obtenus dans ce cas.

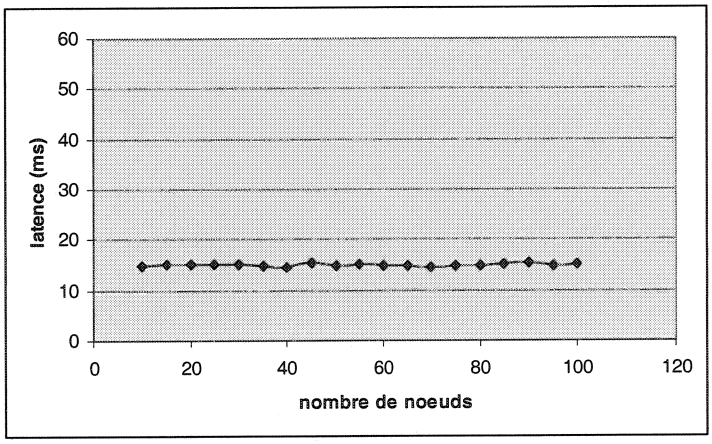


Figure 4.9 : Latence du protocole APM dans le cas où l'accompagnateur possède des adresses libres

On remarque dans ce cas que la latence est très faible et qu'elle ne dépend pas de la taille du réseau, ce qui est tout à fait normal car l'attribution d'adresses se fait de façon presque instantanée.

- **Expérience 2 : cas où l'accompagnateur ne possède pas d'adresses libres.**

Pour accomplir la configuration d'un nouveau nœud, dans ce cas, l'accompagnateur contacte le Nc pour avoir des adresses de configuration. La latence est donnée par la formule (2)

$T_{latence} = T_1 + T_2 + T_3 + T_4 + T_{trait\_acc} + T_{trait\_Nc} + T_{req} + T_{rep}$

Le terme $T_1 + T_2 + T_3 + T_4$ représente le temps d'échange entre n'importe quel nouveau nœud et son accompagnateur. Ce terme est donc indépendant de la taille du réseau, et par conséquent il ne va pas influencer le comportement de la latence vis à vis de la taille du réseau. En outre, le terme : $T_{trait\_acc} + T_{trait\_Nc}$ est négligeables comparativement aux T_{rep} et T_{req}, car les nœuds du réseau ont de bonnes capacités de calcul. De ce fait, on ne tiendra compte, dans le calcul de la latence, que de T_{req} et T_{rep}.

On suppose en plus dans ce cas qu'il n'y a pas de départ du noeud de configuration tout au long de la période de simulation. L'expérience consiste donc à :

1- générer un réseau mobile avec un nombre de nœuds donnés : soit N;
2- désigner au hasard un nœud comme nœud de configuration (Nc);
3- choisir un autre nœud comme accompagnateur, d'un nouveau nœud se trouvant dans son voisinage;
4- mesurer le temps d'envoi d'une demande de configuration de l'accompagnateur au Nc;
5- mesurer le temps de réponse d'une demande de configuration du Nc à l'accompagnateur;
6- recommencer à partir de l'étape 1 en augmentant le nombre de nœuds du réseau.

Cette façon de faire est schématisée sous forme de processus comme le montre la figure 4.10.

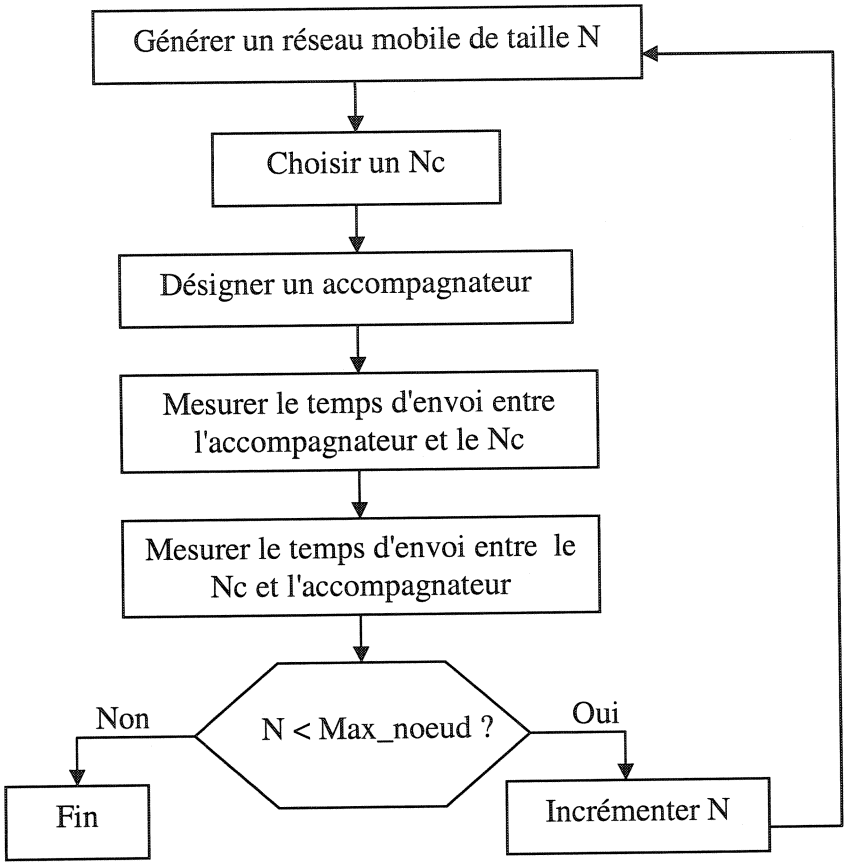


Figure 4.10 : Processus de calcule de latence

N : représente la taille du réseau, qui commence par N = 5.

Max_noeud est la taille maximale du réseau. Dans notre simulation, nous avons travaillé avec Max_noeud = 100.

Nous avons effectué des mesures de latence sur des réseaux de tailles et de topologies initiales différentes. Les résultats obtenus, sont illustrés par la figure 4.11.

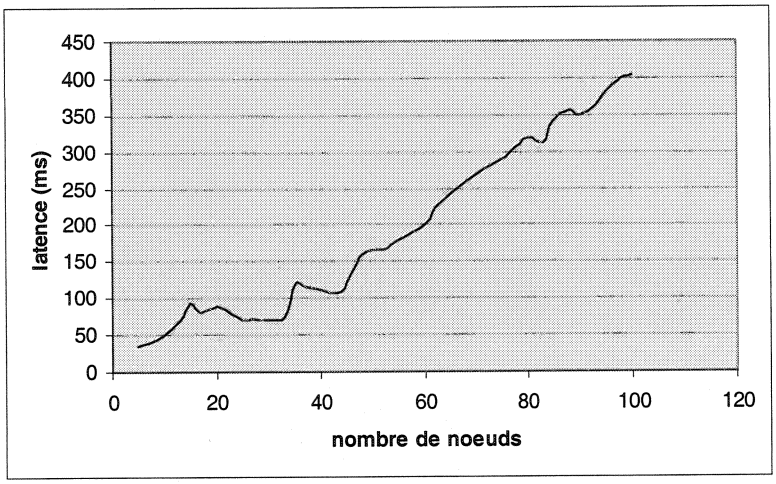


Figure 4.11 : Latence du protocole APM dans le cas où l'accompagnateur ne possède pas d'adresses libres (sans départ du Nc)

La figure 4.11, montre le comportement de la latence en fonction de la taille du réseau. On remarque que le temps pour configurer un nouveau nœud augmente en fonction de la taille du réseau.

- **<u>Expérience 3</u> : cas où l'accompagnateur ne possède pas d'adresses libres et avec départs du nœud de configuration.**

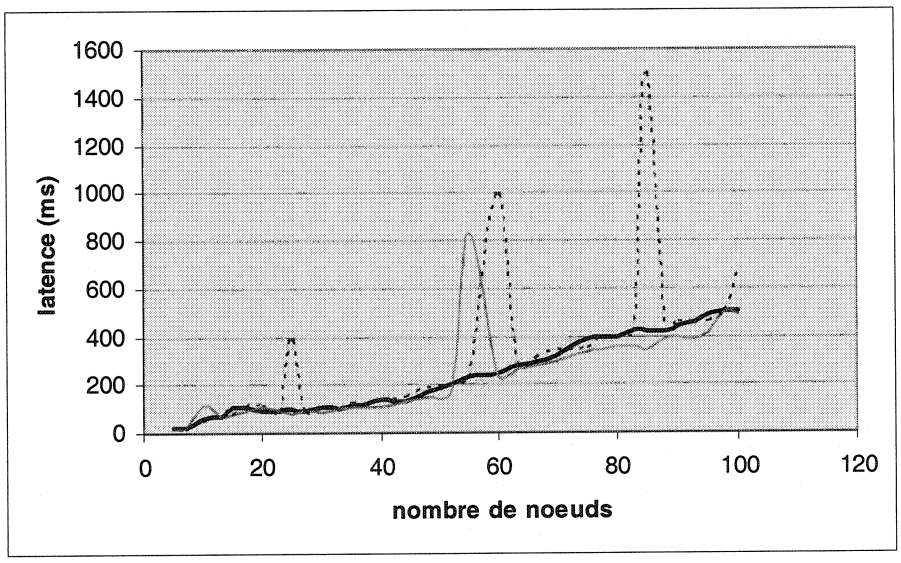
Dans ce cas, nous allons faire des expériences tout en provoquant des départs de nœuds. Cette expérience a été refaite plusieurs fois avec des départs allant de 5 % jusqu'à 35 % de la taille du réseau. Ce point est important car dans le calcul de la latence, il va s'ajouter le temps correspondant au remplacement du Nc. Comme nous l'avons vu précédemment, le délai engendré par un remplacement de Nc est donné par l'équation suivante :

$$TNc_{latence} = T_1 + T_2 + T_3 + T_4 + T_{req} + T_{timer} + T_{trait} + T_{bcast} + T_{rep}$$

Encore une fois, nous n'allons pas tenir compte du temps de traitement (T_{trait}), car les unités mobiles ont de bonnes capacités de traitement. Aussi, le terme $T_1+T_2+T_3+T_4$ sera négligé car il correspond uniquement aux échanges entre le nouveau nœud et son

accompagnateur. De ce fait, nous allons prendre en considération seulement le temps de diffusion (T_{bcast}), le temps d'une demande de configuration (T_{req}), le temps de réponse du réseau (T_{rep}) et le temps d'attente avant de déclencher le mécanisme de remplacement du Nc (T_{timer}) .

Les figures 4.12, 4.13 et 4.14 montrent les résultats obtenus avec différents pourcentages de départs (de 5 % à 35 %).



---------- Latence avec 15 % de départs

_____ Latence avec 10 % de départs

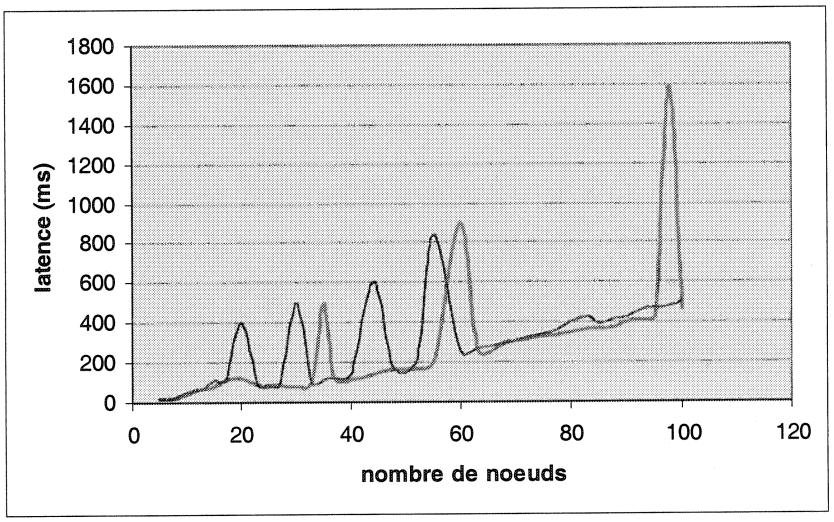
_____ Latence avec 5 % de départs

Figure 4.12 : Latence avec départs du Nc (5%, 10% et 15 %)

Les expériences que nous avons effectué sur des réseaux de taille variant entre 5 noeuds jusqu'à 100 nœuds, montrent que :

- En provoquant 5 % de départs, nous avons constaté que sur les 20 tests effectués, aucun départ du Nc.
- Avec 10 % de départs, nous avons obtenu 1 départ.
- Et, avec 15 % de départs, nous avons constaté 3 départs du Nc.

On constate donc que la latence, des demandes de configurations coïncidant avec un remplacement de Nc, est assez importante, et elle correspond aux pics figurant dans les illustrations de la figure 4.12. Une fois le remplacement du Nc est terminé, les latences deviennent moins importantes.

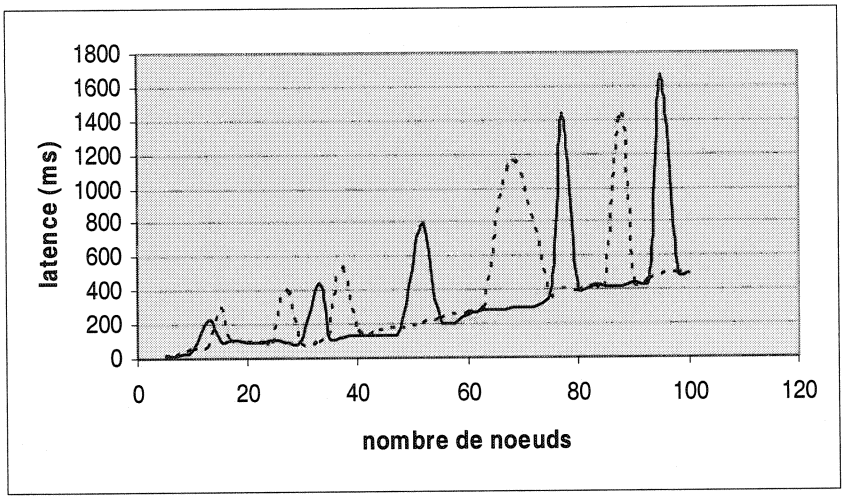


_____ Latence avec 20 % de départs

_____ Latence avec 25 % de départs

Figure 4.13 : Latence avec départs du Nc (20% et 25 %)

Les figures 4.13 et 4.14, montrent aussi le comportement de la latence en présence de départs du nœud de configuration. Les résultas obtenus correspondaient à des provocations de départ allant jusqu'à 35 % de la taille initiale du réseau. On remarque aussi que les départs du Nc deviennent de plus en plus fréquents dépendamment du taux de départs qu'on a fixé au début des expériences.



_____ Latence avec 35 % de départs

---------- Latence avec 30 % de départs

Figure 4.14 : Latence avec départs du Nc (30% et 35 %)

- **<u>Expérience 4</u> : méthode de Nesargi.**

  Comme nous l'avons déjà vu, le temps de configuration d'un nœud est donné par l'équation (4) :

  $$T_{latence} = T_1 + T_2 + T_3 + T_4 + T_{trai\_acc} + T_{bcast} + T_{rep}$$

  Encore une fois, le terme : $T_1 + T_2 + T_3 + T_4$ représente uniquement l'échange entre le nouveau nœud et son accompagnateur, donc il ne va pas influencer le comportement de la latence par rapport à la taille du réseau. Aussi, le terme : T_{trai_acc} sera négligé, en raison de la bonne capacité de traitement des unités mobiles. Donc, pour calculer la latence, on doit mesurer juste T_{bcast} et T_{rep}.

L'expérience dans ce cas consiste à :

1- générer un réseau mobile.

2- choisir un nœud comme accompagnateur.

3- l'accompagnateur diffuse une demande de validation sur tout le réseau.

4- le reste du réseau envoie des réponses à l'accompagnateur.

5- si, toutes les réponses sont affirmatives, l'accompagnateur diffuse un autre message pour informer le réseau de la nouvelle configuration. Sinon, ce processus est relancé.

Les résultats obtenus dans ce cas sont illustrés dans la figure 4.15.

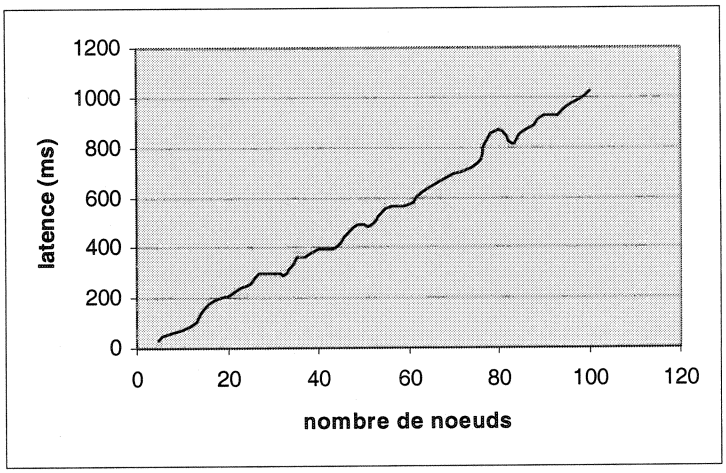


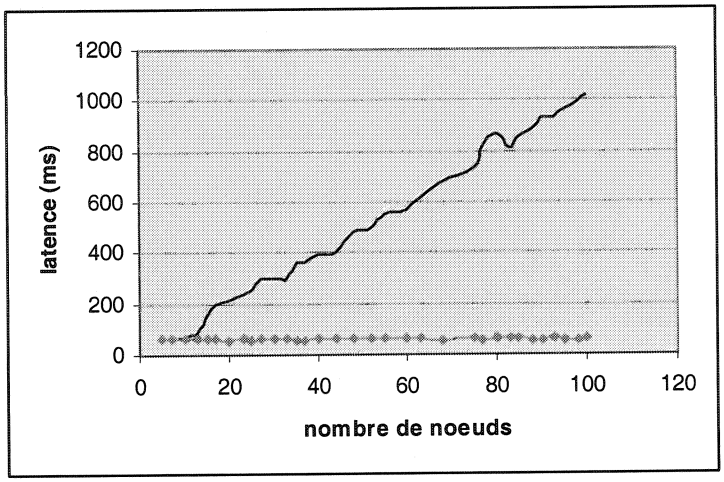
Figure 4.15 : Latence avec la méthode de Nesargi

On remarque que la latence est proportionnelle à la taille du réseau. Dans la section suivante, nous allons comparer de plus près la performance, relative à la latence, du protocole APM et de celui de Nesargi.

4.3.2 Comparaisons basées sur la latence

Dans cette partie nous allons comparer les différents résultats relatifs au délai de configuration d'une nouvelle unité mobile. Ces comparaisons vont concerner les résultats obtenus par le protocole APM et par celui de Nesargi.

Dans la figure 4.16, nous avons représenté à la fois les résultats obtenus par notre méthode de configuration, dans le cas où l'accompagnateur dispose d'adresses libres, et par celle de Nesargi.



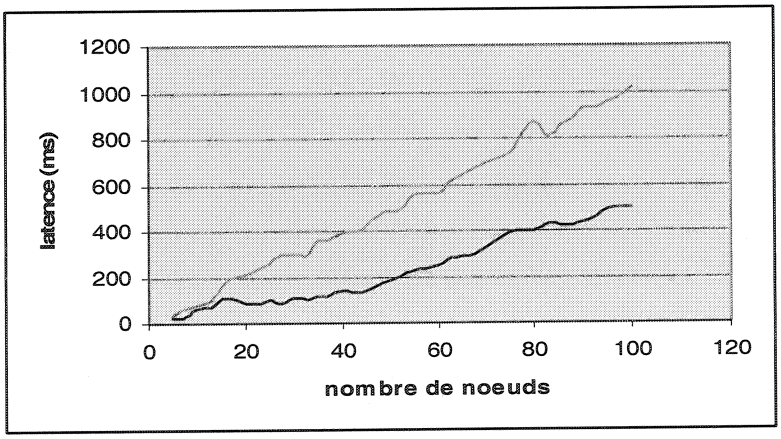
Latence APM

Latence avec la méthode de Nesargi

Figure 4.16 : Comparaison des latences d'APM et Nesargi

La figure ci-dessus montre que la latence du protocole APM dans ce cas, ne dépend pas de la taille du réseau et qu'elle est relativement faible comparativement à celle de la méthode de Nesargi.

Dans la figure 4.17, on montre à la fois les latences obtenues par le protocole de Nesargi et celles obtenues par l'APM dans le cas où l'accompagnateur ne possède pas d'adresses libres.



Méthode de Nesargi

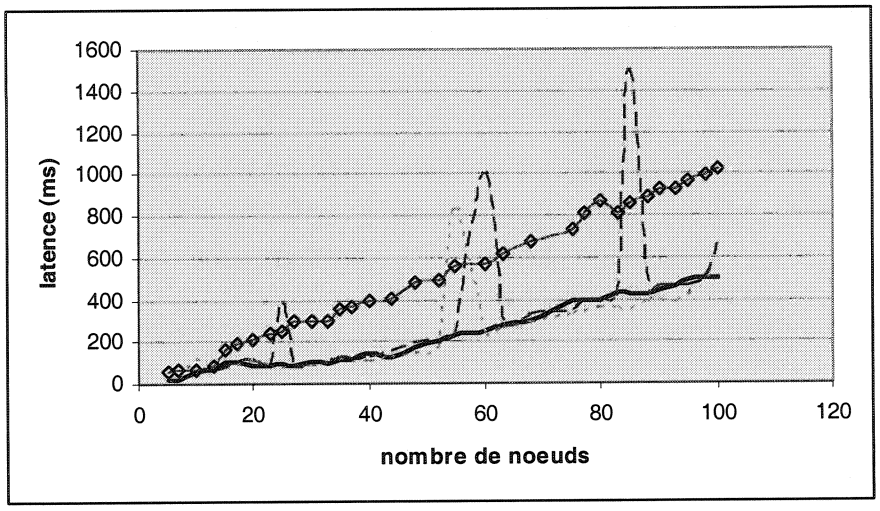
APM

Figure 4.17 : Comparaison des latences en l'absence de départs de noeuds

En faisant abstraction des départs de nœuds, on remarque, comme le montre la figure 4.17, que notre mécanisme de configuration présente de meilleurs résultats en terme de délai de configuration comparativement à la méthode de Nesargi, même si les accompagnateurs qui accomplissent les configurations ne disposent pas d'adresses libres. Dans cette expérience, on a pu noter qu'une configuration par le protocole APM, affiche un gain de temps moyen de l'ordre de 48.06 % par rapport à celui de Nesargi. La question qui se pose à ce niveau est la suivante : quelle sera cette performance en la présence de départs de nœuds? Ce deuxième cas paraît être plus proche de la réalité car les unités mobiles d'un MANET sont très dynamiques. Ce point fera l'objet des prochaines comparaisons.

Pour compléter notre comparaison, nous allons voir dans ce qui suit, que deviendront les performances du protocole APM en présence de départs. Pour se faire nous avons illustré les résultats obtenus en provoquant des départs allant de 5% à 35% de la taille initiale du réseau.

Dans les figures 4.18 et 4.19, nous avons présenté les résultats obtenus en présence de départs dans le réseau mère. Le but des départs provoqués est de s'approcher du comportement observé dans de vrais réseaux. Nous avons répété nos expériences en gardant, à chaque fois, le même pourcentage de départs, mais en faisant varier la taille du réseau de 5 nœuds jusqu'à 100 nœuds. Le nombre de tests réalisés pour une seule expérience peut atteindre le nombre de 30. À force de refaire les tests, des départs de nœuds survient et en conséquence, le comportement de la latence change.



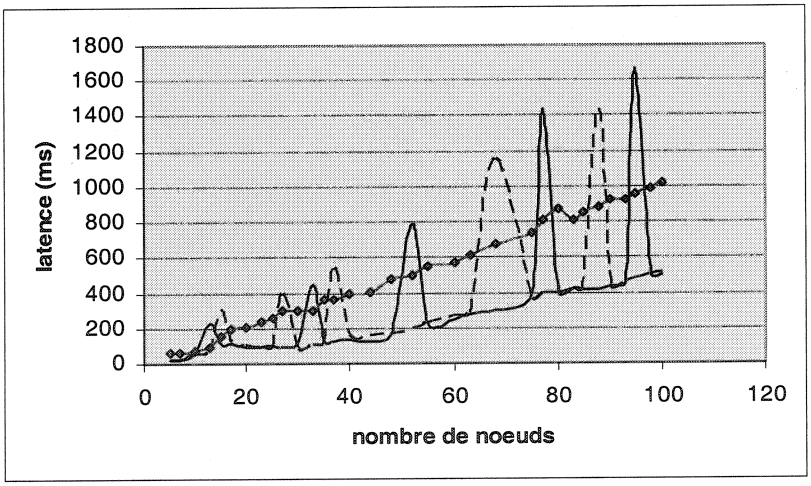
APM avec 5% de départs

APM avec 15 % de départs

APM avec 10 % de départs

Méthode de Nesargi

Figure 4.18 : Comparaison des latences avec des départs de 5% à 15 %



APM avec 35% de départs

Méthode de Nesargi

APM avec 30 % de départs

Figure 4.19 : Comparaison des latences avec des départs de 30% à 35 %

Les cas traités ici concernent, bien sûr, ceux où les accompagnateurs ne possèdent plus d'adresses pour configurer de nouvelles unités mobiles. Dans ce cas, les adresses pour

configurer de nouveaux nœuds vont être cherchées auprès du Nc du réseau. Cependant, en présence de départs, le Nc peut lui aussi quitter le réseau. Dans ce cas, le protocole APM prévoit un mécanisme de remplacement. Les pics qu'on trouve dans les figures 4.18 et 4.19 montrent justement l'impact du départ du Nc sur le délai de configuration. Ce délai, dépasse nettement celui de la méthode de Nesargi et devient de plus en plus fréquent en fonction de la taille du réseau. Pour mieux visualiser l'influence du départ du Nc, nous avons représenté dans la figure 4.20, la différence de latence, en présence de départs du Nc, entre la méthode de Nesargi et le protocole APM.

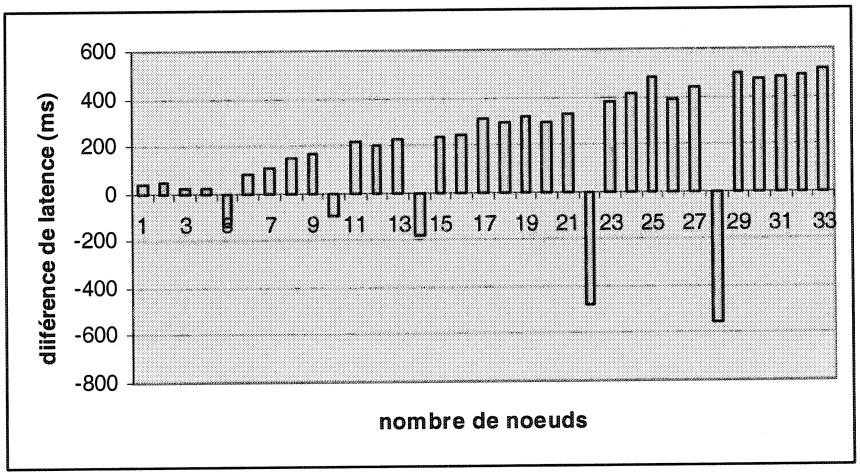


Figure 4.20 : Différence de latence entre la méthode de Nesargi et le protocole APM avec 35 % de départs.

La différence de latence que nous avons présenté dans la figure 4.20 montre qu'avec 35 % de départs de nœuds, une dégradation de performance s'affiche lorsqu'une demande de configuration coïncide avec remplacement du Nc. Cependant, la performance globale du protocole APM reste meilleure et avec un gain moyen de 36.26 % par rapport à la méthode de Nesargi. En plus, nous n'avons pas tenu compte, dans la méthode de Nesargi du cas où un conflit d'adresses est détecté et en conséquence, le mécanisme de configuration doit être relancé.

4.3.3 Expériences relatives aux échanges de messages

Comme dans le cas des délais de configuration, nous allons traiter et commenter les différents cas relatifs au protocole APM et celui de Nesargi.

• **Expérience 1 : Protocole APM**

Les expériences que nous allons réaliser concernent essentiellement les points suivants:

- Cas où l'accompagnateur possède des adresses libres

Les échanges effectués à ce niveau, restent locaux et n'impliquent pas le reste du réseau. En effet, les messages envoyés par un nouveau nœud sont destinés uniquement à ces voisins immédiats. Un nœud est considéré comme voisin s'il arrive à capter son signal. En conséquence, le trafic du réseau en terme de messages ne sera pas influencé. Les messages échangés dans ce cas sont donnés par la formule:

$$N = M_{normal}$$

Où :

M_{normal} réfère au nombre de messages échangés en l'absence de service de configuration.

- Cas où l'accompagnateur ne possède pas d'adresses libres, et sans départ du Nc

L'accompagnateur dans ce cas est obligé de contacter le Nc et d'accomplir un service de configuration. Le nombre de messages échangés dans ce cas est donné par :

$N = M_{normal} + M_{req} + M_{rep}$

Où :

M_{normal} réfère au nombre de messages échangés en l'absence de service de configuration;

M_{req} désigne le nombre de messages pour acheminer une de demande de configuration au Nc;

M_{rep} est le nombre de messages pour recevoir une réponse de configuration.

- Cas où l'accompagnateur ne possède pas d'adresses libres, avec départ du Nc

Dans le cas où une configuration coïncide avec un départ de Nc, le trafic du réseau en terme de message est donné par :

$N = M_{normal} + M_{req} + 2Mbcast + M_{rep}$

Avec :

M_{normal} réfère au nombre de messages échangés en l'absence de service de configuration;

M_{req} est le nombre de messages pour acheminer une de demande de configuration au Nc;

Mbcast désigne le nombre de messages suite à une diffusion dans le réseau;

M_{rep} est le nombre de messages pour recevoir les réponses du reste des nœuds du réseau.

- **Expérience 2 : Protocole de Nesargi**

L'utilisation de la méthode de Nesargi génère des messages sur le réseau. La formule ci-dessous montre le coût de cette méthode de configuration en terme de messages.

$N = M_{normal} + 2Mbcast + M_{rep}$ Où :

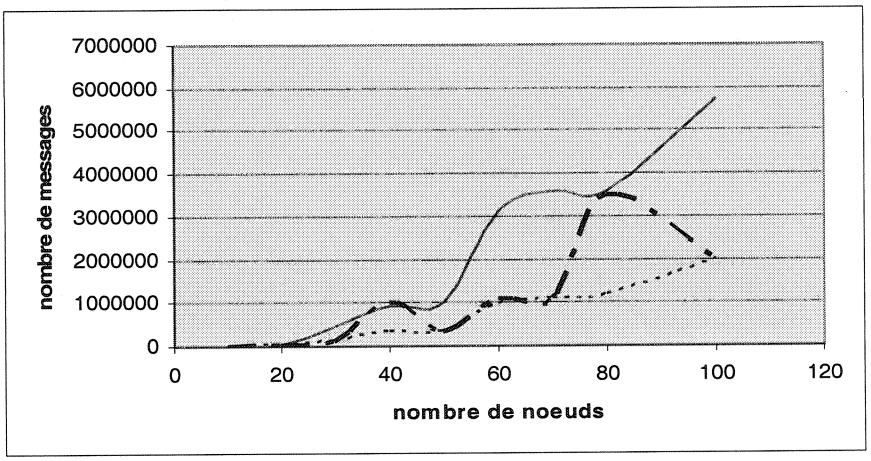
M_{normal} réfère au nombre de messages échangés en l'absence de service de configuration;

Mbcast est le nombre de messages suite à une diffusion dans le réseau;

M_{rep} désigne le nombre de messages pour recevoir les réponses du reste du réseau.

4.3.4 Comparaisons basées les échanges de messages

Après avoir décrit les expériences relatif aux échanges de messages, nous avons effectué des expériences sur des réseaux, de tailles et de topologies initiales différentes. La figure 4.21, montre les résultats obtenus dans les cas où les configurations, avec le protocole APM, se font avec et sans départ du nœud de configuration d'une part, et les résultats obtenus par la méthode de Nesargi d'autre part.



--------- APM sans départs du Nc

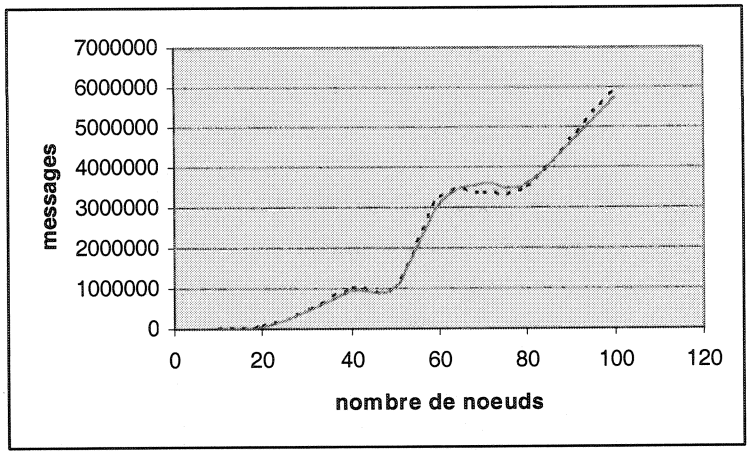
—·—·—·· APM avec départs du Nc

——————— Nesargi

Figure 4.21 : Configuration avec et sans départ du Nc

A partir des résultats de cette figure, on constate que le protocole APM, en présence ou non du départ du Nc, présente de meilleurs résultats en terme de messages. Et ceci sans tenir compte des éventuels relancement du mécanisme de configuration de Nesargi suite à la présence d'un conflit d'adresses.

Supposons maintenant que toutes les configurations que nous allons accomplir avec le protocole APM coïncident avec le départ du Nc. Dans la figure 4.22, nous avons montré les résultats obtenus dans ce cas extrême. On remarque que les configurations avec les méthodes de Nesargi et APM ont le même coût en terme de messages. Donc avec le pire des scénarios, on converge à peine vers la méthode de Nesargi.



———— Nesargi

--------- APM avec départs du Nc

Figure 4.22 : Comparaison de la méthode de Nesargi avec celle d'APM en présence de départs permanents du Nc

4.4 Conclusion

Tout au long de ce chapitre, nous avons pu comparer et discuter des résultats relatifs à la configuration d'une nouvelle unité mobile. Ces résultats concernent les expériences ayant trait au temps de configuration (latence), et celles relatives au trafic. Suite aux résultats obtenus, on peut conclure que le protocole APM, présente de meilleures performances comparativement à celui de Nesargi. Par ailleurs, même en présence massive de départs dans le réseau mère, le protocole APM continue de présenter de bonnes performances. On note aussi, que pendant les expériences menées, nous n'avons pas tenu compte du cas où le protocole de Nesargi échoue. Evidemment, lorsqu'un tel scénario survient, il va se traduire en une augmentation du délai de configuration et du nombre de messages échangés.

Chapitre 5
Conclusion

Dans le présent chapitre, nous allons donner, dans un premier temps une synthèse de notre travail, ensuite nous citerons les limitations de notre proposition de configuration et nous finirons par une présentation des futurs créneaux de recherche relatifs à notre sujet.

5.1 Synthèse des travaux

Tout au long de ce mémoire, il a été question du problème de configuration dans les réseaux mobiles ad hoc. Notre objectif était de proposer un mécanisme d'allocation d'adresses IP aux nouvelles unités mobiles tout en tenant compte de leurs comportements à la fois dynamique et aléatoire. Le protocole d'autoconfiguration pour MANETs (APM) que nous avons proposé assure un service de configuration directe, dans le sens où l'avis du reste du réseau n'est pas sollicité et une gestion centralisée des problèmes émanant de la haute mobilité des composants formant ce genre de réseau. Les problèmes en question concernent particulièrement la gestion des départs, la récupération des adresses non utilisées à l'intérieur du réseau, le partitionnement et la fusion. L'idée du protocole APM est axée sur le fait qu'une nouvelle demande de configuration est adressée aux voisins immédiats du nœud demandeur. Ensuite, un de ces voisins va fournir à la nouvelle unité mobile des adresses IP; s'il en dispose, sinon, il doit les chercher auprès du nœud de configuration. Quant à la supervision et la gestion du service de configuration, elle est assurée par le Nc. Grâce au concept de Nc et à la collaboration des autres nœuds du réseau, le MANET fait preuve d'une intelligence de groupe qu'on retrouve, par exemple, au niveau des décisions à prendre pour réguler le nombre d'adresses libres allouées aux accompagnateurs, aussi au niveau de déclenchement de mise à jour de la table des adresses, etc. Dans le but d'analyser la

performance de notre proposition de configuration, nous avons procédé à des séries de tests moyennant l'outil de simulation GLOMOSIM. Les simulations que nous avons accomplies nous ont permis de mesurer d'une part, le temps nécessaire pour satisfaire une demande de configuration (latence), et d'autre part, le trafic en terme de messages généré suite au lancement du processus de configuration. En guise de comparaison, nous avons implémenté également une méthode de configuration basée sur la détection de conflits, en l'occurrence la méthode de Nesargi. Le choix de cette méthode était justifié par le fait qu'elle est la plus complète des approches de configuration proposées jusqu'à présent. Les résultats obtenus montraient que de point de vue de la latence, le protocole APM présente de meilleures performances même en présence de départs du nœud de configuration. D'autre part, du point de vue trafic, nous avons constaté encore une fois qu'APM génère moins de trafic dans le réseau comparativement au mécanisme de Nesargi. Et même si on considère le cas extrême où on a un départ permanent du Nc, les deux méthodes affichent à peu près le même comportement en ce qui concerne le nombre de messages générés sur le réseau.

5.2 Limitation des travaux

Le protocole APM présente d'importants avantages en terme de latence et de trafic comparativement à la méthode de Nesargi. Cependant, notre proposition de configuration présente aussi des limites. Parmi ces limites on cite le fait de supposer que les unités mobiles du réseau doivent posséder de bonnes capacités de traitement. Cette supposition met une restriction, du moins pour le moment, sur certains équipements qui vont constituer nos prochains réseaux ad hoc. En plus, dans l'élaboration du protocole APM, nous n'avons pas tenu compte des contraintes de sécurité et de perte de messages, ce qui rend l'application de ce genre de mécanisme conditionnée par une utilisation de confiance. Une autre limite concerne le mécanisme de fusion qui, comme nous l'avons présenté dans le chapitre 3, traite ce phénomène uniquement pour deux réseaux. C'est à dire, si plusieurs réseaux se trouvent ensemble dans un même espace géographique, la situation de fusion sera traitée, mais seulement entre deux réseaux à la fois. Ce qui va prendre du temps, dépendamment du nombre de réseaux en état de fusion, pour venir au

bout de toutes les fusions. Bien entendu, ceci ne va pas affecter le fonctionnement normal de chacun des réseaux, mais la formation d'un seul nouveau réseau prendra d'avantage de temps.

5.3 Travaux futurs

Après avoir discuté des limitations de notre travail, nous allons exposer, dans cette section, quelques unes des pistes de recherche ayant un trait direct à notre sujet. Une suite logique de notre travail serait par exemple de traiter le volet sécurité. Plus précisément, ceci revient à ajouter des mesures de sécurité au niveau des composants du réseau, des messages échangés, des traitements, etc. On peut aussi penser à proposer un mécanisme pour choisir les nœuds de configuration sans se préoccuper des capacités de traitement de tous les nœuds du réseau. En d'autres termes, si on dispose d'un tel mécanisme, tous les périphériques mobiles, seront en mesure de se joindre à un MANET. On propose aussi d'améliorer le mécanisme de traitement des fusions de manière à pouvoir détecter et gérer la fusion de plusieurs réseaux à la fois. Un autre point fort intéressant, est celui du "flooding", qui consiste à proposer un outil optimisé pour faire de la diffusion sur le réseau. En effet, dans la revue de littérature, nous avons pu constater que toutes les méthodes proposées dans le cadre de la configuration des réseaux ad hoc, utilisent la diffusion. Ceci est le cas des méthodes de configuration avec détection de conflits, mais les méthodes de configuration sans détection de conflits ont également recours à cette technique pour résoudre les problèmes de partitionnement et de fusion par exemple.

Bibliographie

[Abolhasan 2003], Mehran Abolhasan, Tadeusz Wysocki, Eryk Dutkiewicz "A review of routing protocols for mobile ad hoc networks" Elsevier 2003.

[Cizault 2002] G. Cizault, " IPv6 Théorie et Pratique ", édition O`REILLY, 3éme édition 2002.

[Droms 1997] R. Droms, " Dynamic Host Configuration Protocol", RFC 2131, IETF, March 1997.

[Glomosim 2001] University of California, Los Angeles Department of Computer Science.
http://pcl.cs.ucla.edu/projects/glomosim/

[GSM Radio interface 1996] A brief Overview of the GSM Radio Interface Thierry Turletti Telemedia Networks and Systems GroupLaboratory for Computer ScienceMassachussets Institute of Technology turletti@lcs.mit.eduMarch 1, 1996. http://tns-www.lcs.mit.edu/~turletti/gsm-overview/gsm-overview.html

[IEEE 2003] http://standards.ieee.org/getieee802/

[Kadayif 2004] Kadayif, I.; Kandemir, M, "Tuning in-sensor data filtering to reduce energy consumption in wireless sensor networks ",Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings , Volume: 2 , 16-20 Feb 2004 Pages:852 – 857.

[Krco 2003] Krco, S.; Dupcinov, M, " Improved neighbor detection algorithm for AODV routing protocol ", Communications Letters, IEEE , Volume: 7 , Issue: 12 , Dec. 2003 Pages:584 – 586.

[Martin 1999] J. Martin, " GLOMOSIM Tutorial ".
http://pcl.cs.ucla.edu/slides/workshop99/Jaytut-pw99/index.htm

[McAuley 2000], A. McAuley et al., " Dynamic Registration and Configuration Protocol", draft-itsumo-drcp-00.txt,IETF, July 2000.

[Misra 2001] A. Misra, S. Das, A. McAuley, J.K. Das, "Autoconfiguration, Registration and Mobility Management for Pervasive Computing ", IEEE Personal Communications, Volume 8, Issue 4, Aug 2001, pp 24-31.

[Mohsin 2002] M. Mohsin, R. Parakash, " IP Address Assignment in a Mobile Ad Hoc Network", MILCOM 2002. Proceedings, Volume: 2, 7-10 Oct. 2002 Pages:856 - 861 vol.2, IEEE 2002.

[Nesargi 2002] S. Nesargi, R. Prakash, " Manet conf : Configuration of Host in a Mobile Ad Hoc Network ", INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE , Volume: 2 , 23-27 June 2002 Pages:1059 - 1068 vol.2.

[Nilsen 2001] T. Toftegaard Nilsen, " IPv6 for Future Wireless Networks ", Ericson Telebit, Denmark. 2001 Kluwer Academic Publishers.

[Parsec 1998], Parsec user Manuel,
http://pcl.cs.ucla.edu/projects/parsec/manual/#_Toc428871707

[Perkins 2000] C. Perkins, "IP Mobility Support for IPv4, revised", draft-ietf-mobileip-rfc2002-bis-02.txt, IETF, July 2000, work in progress.

[Perkins 2001] Charles E. Perkins, Jari T. Malinen, Ryuji Wakikawa, Elizabeth M. Belding Royer and Yuan Sun, IP Address Autoconfiguration for Ad Hoc Networks, draft-ietf-manet-autoconf-01.txt, Novembre 2001.

[Samuel 2003] P. Samuel, "Réseaux et Systèmes Informatiques Mobiles, Fondements, architectures et applications ", Polytechnique, 2003.

[Samuel P. 2003] Samuel Pierre, Michel Barbeau, Evangelos Kranakis, " Ad-Hoc, Mobile, and Wireless Networks ", Second International Conference, ADHOC-NOW 2003, Montreal, Canada, October 2003.

[Simpson 1994] W. Simpson, " RFC 1661 - The Point-to-Point Protocol (PPP)
", Network Working Group Request for Comments: 1661, Daydreamer
STD: 51 July 1994, Category: Standards Track.
http://www.faqs.org/rfcs/rfc1661.html

[Soo 2001] Jung Soo Park, Yong Jim Kim and Sung Woo Park, " Stateless Address Autoconfiguration in Mobile Ad Hoc Networks Using Site-Local Address ", 2001 Internet Draft : draft-park-zeroconf-manet-ipv6-00.txt.

[Ulas 2003]Ulas C. Kozat' and Leandros Tassiulas, " Service discovery in mobile ad hoc networks: an overall perspective on architectural choices and network layer support issues " Department of Electrical and Computer Engineering and Institute for Systems Research, University of Maryland, College Park, MD 20742, USA Received 5 May 2003; accepted 30 June 2003. ; Available online 29 August 2003.

[Weniger 2002] K, Weniger, M. Ztterbart, " IPv6 Autoconfiguration in Large Scale Mobile Ad Hoc Networks ", in : Proceedings of European wireless 2002, Florence, Italy, Feb 2002.

[Wilson 1995] R Paul Wilson, S. Johnstone, M. Neely, D. Boles, " Dynamic Storage Allocation: A survey and critical review", International Workshop on Memeory Management, September 1995.

[Zhou 2003] H. Zhou, M. Lionel, Matt W. Mutka, " Prophet address allocation for large scale MANETs", INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE , Volume: 2 , 30 March-3 April 2003

Pages:1304 - 1311 vol.2.

[Zhu 2003] Chunhui Zhu, Myung J. Lee, Tarek Saadawi, "A Border-aware Broadcast Scheme forWireless Ad Hoc Network", Electrical Engineering Dept, Engineering School

City University of New York, CCNY, New York, USA, IEEE 2003.