Titre: Title:	Réseau dorsal virtuel pour la découverte de services dans les réseaux ad hoc
Auteur: Author:	Ranwa Al Mallah
Date:	2008
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Al Mallah, R. (2008). Réseau dorsal virtuel pour la découverte de services dans les réseaux ad hoc [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8241/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/8241/
Directeurs de recherche: Advisors:	Alejandro Quintero
Programme: Program:	Unspecified

UNIVERSITÉ DE MONTRÉAL

RÉSEAU DORSAL VIRTUEL POUR LA DÉCOUVERTE DE SERVICES DANS LES RÉSEAUX AD HOC

RANWA AL MALLAH
DÉPARTEMENT DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

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

AVRIL 2008

Your file Votre référence
ISBN: 978-0-494-41546-7
Our file Notre référence
ISBN: 978-0-494-41546-7

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

RÉSEAU DORSAL VIRTUEL POUR LA DÉCOUVERTE DE SERVICES DANS LES RÉSEAUX AD HOC

présenté par : AL MALLAH Ranwa

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

M. PIERRE Samuel, Ph.D., président

M. QUINTERO Alejandro, Doct., membre et directeur de recherche

M. FERNANDEZ José M., Ph.D., membre

REMERCIEMENTS

Je remercie mon Directeur de recherche, M. Alejandro Quintero pour ses critiques, ses conseils et son soutien indéfectible tout au long de ce travail.

Je remercie également M. Rawad pour sa grande disponibilité et ses précieux commentaires pour la rédaction et la correction de ce mémoire.

Je remercie aussi mes parents pour leurs encouragements et mon mari pour son support tant désiré.

Je remercie enfin les membres du LARIM pour leur collaboration et leur soutien.

RÉSUMÉ

Dans les réseaux ad hoc communément appelés MANET (*Mobile Ad Hoc Network*), bien que le routage soit d'une grande importance, il existe également un autre élément d'intérêt qui est la déclaration et la découverte de service. La découverte de service est le processus par lequel un utilisateur, une application ou un terminal peuvent localiser, récupérer de l'information et finalement utiliser un service disponible dans le réseau ambiant. Un service est une entité qui peut être utilisée par une personne, un logiciel, un canal de communication, un dispositif, ou autre.

En effet, les unités mobiles n'ont pas la possibilité d'accéder facilement à l'ensemble des services offerts par le réseau et ceci à cause de leurs ressources limitées (bande passante faible, énergie limitée, faible portée radio, puissance de calcul faible, mémoire restreinte). Dans un environnement maîtrisé, la question est vite réglée : l'ensemble des composants logiciels est connu, restreint et les mises à jour centralisées. Parallèlement, les propositions de découverte de service pour les réseaux ad hoc fournissent des algorithmes permettant une décentralisation totale de la construction et de la maintenance de systèmes distribués.

L'objectif de ce mémoire consiste à concevoir et développer un modèle pour la découverte de service en tenant compte des contraintes auxquelles sont soumises les unités mobiles. De façon spécifique, ce modèle utilisera un mécanisme qui différencie les nœuds stables des nœuds instables pour former un sous-ensemble de nœuds constituant un réseau dorsal virtuel (RDV). En effet, il s'agit de former un RDV même dans un état du réseau où la mobilité est très élevée en désignant des unités mobiles clés pour héberger la liste des services offerts dans le réseau. La stratégie de déploiement d'un RDV pour repérer et enregistrer les services à l'intérieur d'un réseau dont la topologie change dynamiquement est apparue très prometteuse.

Pour évaluer la performance de la solution proposée, des simulations ont été réalisées sur différents scénarios, avec en sortie trois indices : la charge réseau, le taux moyen de succès et le délai moyen de réponse aux requêtes. Il en ressort que le modèle

génère des résultats très satisfaisants quant à la charge du réseau en termes de nombre de paquets de contrôle. Le pourcentage de diminution du nombre de messages de signalisation est de plus de 80% comparativement aux meilleures propositions dans la littérature. De plus, le taux moyen de succès aux requêtes a subit une amélioration de près de 1% en moyenne sur l'ensemble des cas considérés. Amélioration minime mais considérant que les stratégies considérées avaient déjà de très bons taux de succès de l'ordre de 90% de tentatives réussites, nous pouvons marquer ce perfectionnement. Les délais moyens de réponse aux requêtes sont raisonnables mais perfectible.

Au final, le modèle proposé ouvre une piste de recherche très prometteuse dans des environnements aussi décentralisés et élargies que sont les réseaux MANET.

ABSTRACT

In mobile ad hoc networks, although routing is of great importance, another element of interest is service discovery. Service discovery is the process by which a user, an application or a terminal can locate information and finally use an available service in the surrounding environment. A service is an entity that can be used by a person, software, a communication channel or a device.

Indeed, the mobile units often do not have the possibility of easily reaching all the services offered by a network because of their limited resources (low bandwidth, limited energy, short radio range, low computing power, restricted memory). In a controlled environment, the problem is solved: all software components are known and updates are centralized. On the other hand, the proposals for service discovery in mobile ad hoc networks provide algorithms allowing a complete decentralization for construction and maintenance of distributed systems.

The objective of this thesis is to conceive and develop a model for service discovery by taking into account the limited resources to which the mobile units are subjected. More specifically, the model will use a mechanism which differentiates stable from unstable nodes in the network in order to form a subset of nodes constituting a virtual backbone (VBB). In fact, our approach is to form a VBB even in a network with increased mobility by selecting key mobile units to store the list of services offered by the network. The VBB deployment strategy to spot and record services inside a network with frequent topology changes appeared very promising.

To evaluate the performance of the proposed solution, we implemented it and simulated through a series of simulations, by having the focus on the network load, the hit ratio and the average delay to requests. It emerges that the model generates very satisfactory results as for the network load in terms of number of control packets. When compared with other strategies, the percentage of reduction of signalling messages is more than 80%.

Our results show that the proposed model outperforms the other strategies, in terms of network load, hit ratio and average delay to requests.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

SIGLES, ABRÉVIATIONS, ACRONYMES

RDV	Réseau Dorsal Virtuel
BB	Backbone
NLFF	Normalized Link Failure Frequency
NIT	Network Information Table
DSDP	Disrtributed Service Discovery Protocol

CHAPITRE I

INTRODUCTION

Les micro-ordinateurs d'aujourd'hui exécutent plusieurs fois plus d'instructions par seconde que ce que pouvaient faire de gros ordinateurs à l'époque. La progression des télécommunications est encore plus extraordinaire. Il y a quarante ans les réseaux de haute performance communiquaient à 54Kbps; les liens modernes sur fibre optique transmettent au-delà de 10 Gbps. Depuis les premiers jours de l'informatique, le rapport entre la taille et la puissance des ordinateurs décroit. Des études rétrospectives montrent qu'une limite a été atteinte il y a près de deux décennies, avec le lancement des ordinateurs personnels. Le marché des ordinateurs de poche croit très rapidement et, en plus des avantages en termes de taille, poids et coût qu'ils offrent, ces ordinateurs portables devront pouvoir interagir entre eux.

D'autre part l'augmentation du nombre de portables nécessite un support réseau approprié qui permette de fréquents changements de topologie. Le terme « réseaux ad hoc » désigne de tels réseaux qui sont capables de se reconfigurer automatiquement et rapidement. La poursuite de la réduction de taille et de coût des portables nécessite que ceux-ci soient dotés d'une capacité de communiquer directement entre eux, afin de mieux partager les ressources réduites. Ainsi, les ordinateurs individuels ont rendu possible l'introduction des applications bureautiques, la portabilité a atteint un niveau suffisant pour permettre l'émergence de nouvelles applications et l'évolution des services dans le réseau. Face à cette évolution, il paraît nécessaire de pouvoir déclarer, découvrir et déployer des services de la façon la plus transparente possible et surtout dynamiquement. Il nous semble intéressant de se pencher sur le cas des réseaux ad hoc car ce domaine de l'informatique mobile est en plein essor. Dans ce chapitre d'introduction, nous exposerons de prime à bord les éléments de la problématique de déclaration et découverte de services dans les réseaux ad hoc. Nous enchaînerons avec les objectifs de la recherche et nous terminerons avec une esquisse du plan de mémoire.

1.1 Définitions et concepts de base

Un réseau ad hoc est une communauté de terminaux mobiles communicants via des liens sans-fil sans le support d'une infrastructure. En conséquence, les unités mobiles forment un réseau temporaire abrogé d'administration pour sa gestion ou configuration. Dans le plus simple des cas, les mobiles se trouvent dans la zone radio les uns des autres et forment un réseau ad hoc à un saut. Ce qui est plus intéressant sont des terminaux qui ont besoin de nœuds intermédiaires pour communiquer les uns avec les autres. Dans ce scénario, les nœuds intermédiaires prennent la fonctionnalité de routeurs classiques. Dans un réseau ad hoc, les nœuds interagissent ensemble en tant qu'égaux et sont caractérisés par :

- une topologie qui peut changer aléatoirement et rapidement, les nœuds sont libres de se déplacer;
- aucune infrastructure préexistante, aucune administration centralisée;
- un nœud est à la fois routeur et station hôte; acheminement autonome de paquets d'un usager à un autre;
- contrainte d'énergie car le nœud est alimenté par une source d'énergie autonome;
- la portée de communication est limitée;
- la durée de vie des liaisons est limitée dans le temps, du fait de la mobilité et de la consommation d'énergie.

Ces caractéristiques écroulent la majorité des protocoles mis en place dans les réseaux filaires pour en développer de nouveaux qui tiennent compte de la méthode d'accès ad hoc du réseau. Entre autres, les protocoles de routage doivent être adaptés pour router convenablement les messages à travers le réseau dont la topologie change fréquemment. De plus, étant donné la réserve d'énergie restreinte, le nombre de messages qu'un terminal peut traiter et en particulier transférer est limité. La découverte de services est aussi un processus qui doit être redéfini pour les réseaux ad hoc. La découverte de service est le processus par lequel un utilisateur, une application ou un terminal peuvent localiser, récupérer de l'information et finalement utiliser un service

disponible dans le réseau ambiant. Un service est une entité qui peut être utilisée par une personne, un logiciel, un canal de communication, un dispositif, ou autre. Ainsi, la découverte de services fait intervenir deux concepts qui valent d'être explicités :

- requête de service : principe désigné par la découverte de service quand un nœud sollicite ou cherche dans son milieu un service donné. Le nœud formule une requête de service pour explorer la disponibilité d'un service dans le réseau ambiant;
- annonce de service : principe désigné par la découverte de service quand un nœud désire annoncer ou publier dans le réseau le service qu'il offre. Le nœud formule une annonce de service pour avertir son environnement du service qu'il propose.

Dans les réseaux filaires, la découverte de services est résolue au niveau de l'application où un répertoire centralise les informations sur les services. Dans ces réseaux fixes, un seul équipement agit en tant que répertoire de services et se charge de classer la liste des services offerts dans le réseau. Cet équipement est sollicité soit par l'utilisateur qui réclame un service ou parallèlement, par le client désirant un service.

Les évolutions en matière d'informatique mobile ont conduit à des exigences de reconfiguration dynamique des systèmes distribués. Toutes ces stratégies à l'origine utilisées pour les réseaux filaires, doivent être réadaptées aux réseaux ad hoc dont l'architecture complètement distribuée est basée sur des entités mobiles servant de routeurs les unes aux autres et requérant des accès aux informations indépendamment de leur emplacement. Ceci a donné lieu à la définition de différents protocoles pour la découverte de services. Par ce fait, les protocoles de découverte de services se présentent sous la forme de services de courtage qui gèrent des bases de données de services et qui traitent des requêtes. Dès lors que l'on souhaite permettre aux usagers d'un réseau de découvrir aisément les services applicatifs qui peuvent leur être fournis, il est nécessaire soit, de centraliser les services sur un équipement – ou un ensemble d'équipements – précis, ou de disposer de mécanismes permettant d'annoncer au sein du réseau la présence des services, services qui peuvent être alors répartis.

Pour un réseau mobile conçu comme un prolongement d'un réseau d'infrastructure, la centralisation des services applicatifs se révèle être une solution pertinente en termes de facilité de gestion et de découverte des services. En revanche, dans les réseaux mobiles sans infrastructure capables de se former dynamiquement, insinuant les réseaux ad hoc, il n'existe pas d'équipements privilégiés sur lesquels on puisse centraliser les services applicatifs. Considérant qu'il n'est pas judicieux de désigner des équipements particuliers pour jouer de rôle de serveur de services dans de tels réseaux, le problème est alors de définir comment trouver les services dans un environnement mobile.

1.2 Éléments de la problématique

L'exploitation des supports de communication multiples permet d'envisager l'ubiquité des services. Elle est à la base des réseaux de 4e génération. Le modèle ad hoc permet d'étendre la couverture des réseaux d'accès sans fil (802.11) en mode de base et l'utilisation conjointe de ces modes permet outre l'extension de la connectivité, la création de services novateurs dont les domaines d'application vont des réseaux embarqués aux réseaux domestiques en passant par les espaces intelligents. En raison de leur potentiel important, les réseaux ad hoc méritent d'être étudiés et une des problématiques soulevée par les experts est la découverte de services. En effet, les mécanismes actuels de découverte de services manquent de fonctionnalités et deviennent inefficaces dans des environnements dynamiques à large échelle. Il est donc indispensable de proposer de nouveaux outils pour de tels environnements. Le problème que l'on cherche à résoudre est de permettre une découverte flexible (recherche multicritères, complétion automatique) des services dans un environnement dynamique à large échelle en tenant compte de la topologie du réseau physique sous-jacent.

Ainsi, dans les réseaux filaires, la découverte de services est résolue au niveau de l'application où un répertoire centralise les informations sur les services. Dans ces réseaux un service est représenté par un agent de service et un utilisateur opère au nom

d'une application client. Des propositions pour la découverte de services dans les réseaux ad hoc ont essayé de transposer cette idée de répertoire centralisé. Cependant, dans les réseaux ad hoc, les approches qui font appel à un répertoire centralisé ne s'appliquent pas pour les raisons qui suivent :

- en raison de leur nature mobile, les nœuds peuvent joindre ou quitter un réseau à n'importe quel instant donné, ainsi, les services offerts dans le réseau changent fréquemment rendant difficile la mise à jour du répertoire de services;
- dans ce genre de réseau, il n'existe aucune garantie qu'un nœud pourrait être disponible pour une certaine période de temps. Ainsi, aucun des nœuds participants au réseau ne peut servir de répertoire de services;
- il faut prendre en compte la spécificité de ces réseaux : la bande passante limitée, le changement dynamique de la topologie en fonction du temps ainsi que le manque d'informations complètes sur l'état du réseau;
- les déconnexions fréquentes des usagers, c'est-à-dire que les terminaux ne restent pas indéfiniment connectés, ceci dans le but d'économiser l'énergie déjà limitée dont dispose leur batterie;
- la communication entre les stations mobiles étant par voix radio, la qualité du lien sans-fil reste inconnue et susceptible à des variations suivant la configuration et l'état du réseau, offrant par ce fait, une qualité de service imprévisible.

Chacune de ces raisons a un impact direct sur la manière dont devrait être conçu le protocole de découverte de service approprié. Le motif de déconnexion des unités mobiles a une influence indéniable sur la manière dont les communications doivent s'effectuer entre entités du réseau ad hoc. En effet, étant donné que les services dans le réseau changent fréquemment, il y a donc une nécessité de définir une procédure de mise à jour du répertoire de service qui soit la plus transparente possible.

Les solutions existantes utilisées dans les réseaux filaires où les usagers sont toujours connectés, ne pourraient pas convenir à un environnement aussi dynamique que

les réseaux ad hoc. En effet, il n'existe aucune garantie qu'un nœud pourrait être disponible pour une certaine période de temps. Ainsi, le principe de données entreposées sur un seul équipement pose un problème d'envergure. Si toutes les unités mobiles doivent effectuer à chaque fois des requêtes en direction d'un seul répertoire de services, cela risquerait de surcharger le serveur en question ainsi que d'augmenter la latence d'accès à l'information requise. Si à un certain instant le nombre d'unités devient important, il n'y aurait plus aucune garantie de service de la part du serveur, d'où le problème d'évolutivité.

Certaines des propositions de découverte de service se sont focalisées sur la décentralisation du répertoire de services sur plusieurs entités mobiles susceptibles d'agir en tant que serveur. Par ce fait, toutes les unités ayant les spécifications requises sont en mesure de stocker la liste de services offerts dans le réseau. La réplication du répertoire de services sur différentes entités est originale mais vient relever une autre complication. Si toutes les unités mobiles ayant la capacité requise doivent disposer d'une mémoire pour stocker les mêmes données dans cette dernière, cela révèlerait d'un gaspillage d'espace mémoire. Les principaux problèmes posés, qui ont trait au gaspillage de ressources, ne trouvent pas encore de solutions satisfaisantes et font pour la plupart l'objet de différentes études. Le principe de décentralisation ouvre des perspectives nouvelles mais une question reste en suspens : quelle est la meilleure décentralisation à considérer dans un tel milieu ayant des caractéristiques aussi particulières qu'attrayantes?

1.3 Objectifs de la recherche

L'objectif principal de ce mémoire est de proposer une approche pour la découverte de services dans les réseaux ad hoc. Le défi est de permettre aux terminaux de découvrir les services applicatifs qui sont spontanément offerts par les équipements partageant leur médium de communication. Également, il s'agit de permettre à ces terminaux d'annoncer les services qu'ils souhaitent rendre accessible aux membres du

réseau, tout ceci en minimiser les messages de signalisation dans le réseau. Les objectifs de la recherche sont donc les suivants :

- étudier les propositions déjà apportées à ce niveau afin d'en souligner les faiblesses;
- proposer et concevoir un protocole de découverte qui aura la capacité de notifier les services disponibles, découvrir automatiquement les services voisins et ceux qui ne le sont pas, discuter les capacités du service tout en minimisant les messages de signalisation à travers le réseau;
- simuler le protocole proposé afin d'en étudier le fonctionnement;
- évaluer la performance du protocole à l'aide d'indices de performance qui sont des descripteurs de l'efficacité de la proposition par rapport aux meilleurs modèles répertoriés dans la littérature;
- analyser les résultats de simulation obtenus par rapport aux attentes de la proposition.

1.4 Plan du mémoire

La suite de ce mémoire s'articule de la manière suivante. Le chapitre II offre une vue d'ensemble des travaux dont la démarche et la finalité s'apparentent à la nôtre. Ainsi, nous allons définir quelques concepts de base relatifs à la découverte de services dans les réseaux ad hoc pour ensuite faire état d'une revue de littérature à ce propos.

Le chapitre III introduira la proposition et son principe de fonctionnement en présentant les fonctionnalités de découverte et d'annonces des services applicatifs au sein des réseaux ad hoc qui ont la propriété fondamentale d'être extrêmement dynamiques.

Nous présenterons dans le chapitre IV tous les éléments ayant servis à l'implémentation de la proposition. Nous décrirons les étapes menant à la simulation et à l'analyse de performance. Cette dernière nous permettra d'effectuer une comparaison des indices de performance par rapport aux autres protocoles proposés dans la littérature.

Dans un dernier chapitre, nous présenterons une synthèse des résultats du mémoire en mettant en lumière les principales contributions apportées par notre approche. Il s'agira aussi de conclure en présentant des travaux futurs relatifs à la découverte de service proposée.

CHAPITRE II

DÉCOUVERTE DE SERVICES DANS LES RÉSEAUX AD HOC

Un mécanisme efficace de découverte des services est essentiel dans un environnement spontané. Cet environnement offre aux équipements connectés par réseau sans fil un moyen d'intégration à l'intérieur d'un réseau de communication global. Il assiste de manière transparente ces appareils et leur fournit des fonctions avancées. Les communications mobiles ad hoc compliquent le recours à la fonction de découverte des services à cause des restrictions des ressources et de la dynamique de la topologie engagée dans le réseau. Les mécanismes actuels de découverte de services deviennent inefficaces dans des environnements dynamiques à large échelle. Il est donc indispensable de proposer de nouveaux outils pour de tels environnements. Les modèles émergents fournissent des algorithmes permettant une décentralisation totale de la construction et de la maintenance de systèmes distribués performants et tolérants aux pannes.

Le problème que l'on cherche à résoudre est de permettre une découverte flexible (recherche multicritères, complétion automatique) des services prenant place dans un environnement dynamique à large échelle en tenant compte de la topologie du réseau physique sous-jacent. Pour comprendre cette problématique de la découverte de service dans un réseau ad hoc, nous commencerons par distinguer les solutions proposées pour un réseau filaire et un réseau sans-fil à courte portée. Ensuite, nous aborderons les protocoles d'envergure proposés pour des réseaux de large densité. Mais avant, nous ferons un bref survol des réseaux ad hoc.

2.1 Réseaux ad hoc

Par conception, un réseau ad hoc est très peu structuré: il s'agit de nœuds, tous équivalents du point de vue du réseau, qui doivent se charger eux-mêmes d'établir toute l'infrastructure permettant les communications. Pour communiquer, un nœud ne peut émettre que si tous ces voisins écoutent car dans les WLAN, les hôtes se partagent les fréquences, d'où les risques de collision. De plus, les réseaux Manet sont très souvent constitués de nœuds ayant des ressources limitées. En effet, la portée des communications, qui conditionne la puissance d'émission, a une forte influence sur l'autonomie électrique. La capacité de calcul et la bande passante sont autant de contraintes d'utilisation. Les réseaux ad hoc nous obligent ainsi à prêter particulièrement attention à l'utilisation de leurs ressources.

Les champs d'application de ces réseaux sont divers : déploiement d'un réseau durant une opération militaire sur un champ de bataille ou durant une opération de sauvetage dans un lieu difficilement accessible, etc. Quelle que soit l'application visée, un réseau Manet possède des exigences spécifiques du fait de ses particularités. Jusqu'à présent, de nombreux travaux traitent de la sécurisation des réseaux ad hoc [1], des protocoles de routage [2], des mécanismes de découverte de services [3] et bien d'autres. Cependant, différents points restent non abordés, ou insuffisamment traités dans les travaux existants.

En particulier, il s'agit de définir un protocole de découverte de service adapté aux réseaux ad hoc mobiles de grande étendue; il nous paraît nécessaire de traiter les réseaux ad hoc de manière plus globale et en tenant compte des spécificités de tels réseaux. La difficulté est de proposer des mécanismes relativement robustes, sans pour autant affecter les performances du réseau ad hoc de manière trop prononcée.

2.1.1 Requis d'un protocole de Manet

Un réseau Manet est un système autonome de nœuds mobiles reliés par des liens sans fils dont l'union forme un graphe arbitraire. Tel que déjà invoqué, les nœuds du

réseau jouent le rôle de routeurs et sont libres de se déplacer aléatoirement et de s'organiser arbitrairement. En conséquence, la topologie du réseau peut changer rapidement et de manière imprévisible. Un tel réseau ne nécessite pas d'infrastructure fixe et représente une option attractive pour connecter spontanément des terminaux mobiles. Dans un réseau Manet, moins il y a de transmission, moins il y a de congestion dans le réseau. En résumé, pour être efficace, un outil de Manet doit avoir les caractéristiques qui suivent :

- mécanisme de signalisation efficace : étant donné la contrainte de largeur de bande, le mécanisme doit faire en sorte que les liens ne soient pas encombrés avec le trafic de gestion;
- architecture légère: les nœuds ont des ressources limitées mais des grands besoins de traitement et d'auto organisation;
- automatisme, intelligence et flexibilité : étant donné la nature dynamique, une gestion adaptative qui réagit aux changements;
- échange sécuritaire des données de gestion afin d'augmenter la survie globale du réseau.

Subséquemment, pour être convenables, les solutions proposées pour la découverte de service dans un réseau Manet nécessitent d'avoir recours à des méthodes ayant les caractéristiques évoquées plus haut. Les caractéristiques intrinsèques des réseaux ad hoc requièrent donc d'une stratégie qui tienne compte des différents points cités.

2.2 Découverte de service

La localisation des services plus connue sous la dénomination découverte des services, illustrée à la Figure 2.1, a 3 fonctionnalités principales : la publication d'un service, la recherche d'un service et la mise en relation du client avec un serveur d'application. La recherche de service est un composant logiciel implémentant une méthode ou un ensemble de méthodes permettant de sélectionner un service sur la base des paramètres requis par un requérant. Elle consiste principalement à sélectionner un

service désiré, puis de trouver le serveur d'applications le mieux approprié aux requis de la requête envoyée. Elle retourne au requérant l'adresse du serveur d'applications et le met en relation avec le serveur d'applications sélectionné grâce à une redirection de la requête vers l'adresse dudit serveur.

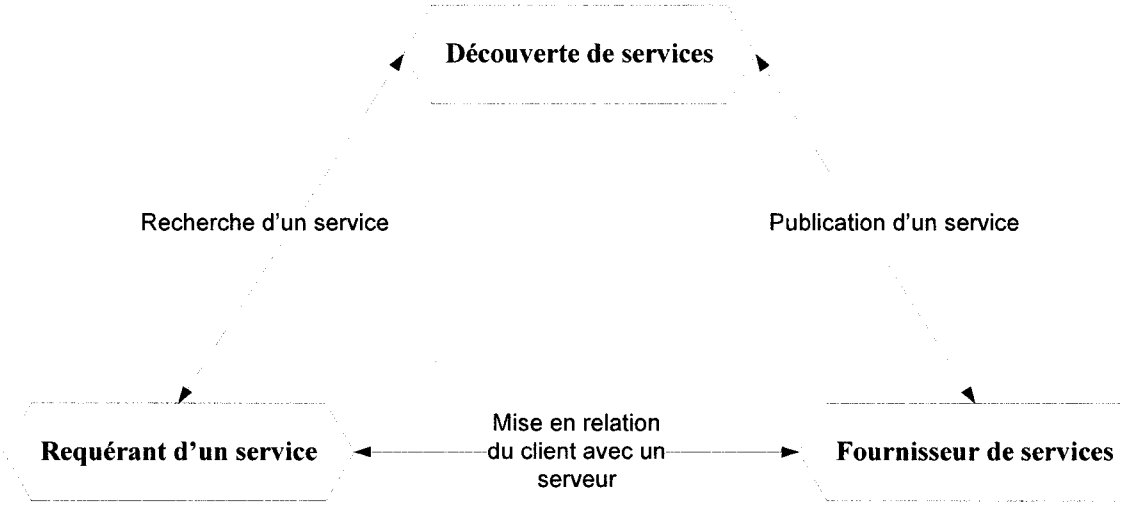


Figure 2.1 Processus de découverte de service

La publication de service est aussi un composant logiciel implémentant un ensemble de méthodes permettant de diffuser un service qu'un fournisseur offre. Elle consiste notamment à formuler une offre détaillée du service à publier pour ensuite annoncer sa disponibilité à travers le réseau. La découverte de services permet ainsi à deux dispositifs de communiquer l'un à l'autre leurs fonctions. Elle ne doit pas se limiter à publiciser une certaine classe de dispositifs ou les fonctionnalités des services. Ce n'est plus suffisant de trouver une imprimante, c'est nécessaire de connaître aussi les détails de cette imprimante : les résolutions, la couleur, la quantité de papier, etc.

2.2.1 Définition d'un service

Un service est une entité qui peut fournir des informations, effectuer une action ou contrôler une ressource. Il peut être implémenté comme un logiciel, du matériel ou comme une combinaison des deux. Un service peut être un serveur Web, une imprimante, une application de diffusion de flux vidéo à la demande, etc. Afin de proposer un protocole de découverte de service, on remarque qu'il faut d'abord que les terminaux offrants les services s'entendent sur la description du service en question. Décrire un service est donc essentiel à son utilisation.

2.2.2 Description d'un service

Sans une description associée, on ne peut pas savoir ce qu'un service propose. Il faut qu'un service publie ses caractéristiques de manière claire. En fait, cette description peut être normalisée ou pas: soit des champs connus sont limités à des valeurs prédéfinies, soit le concepteur du service est entièrement libre d'en définir sa description. La description pourrait être articulée autour de langages tels XML 3, DAML 4 ou WSDL 5, qui permettent des descriptions précises des objets.

Il existe plusieurs solutions pour la description des services. Une des méthodes les plus simples consiste en une page Web qui décrit le service et fournit un lien vers un fichier WSDL. Une autre possibilité est d'utiliser WSIL (Web Services Inspection Language), ou mettre en œuvre un registre UDDI privé. Cependant, la plupart des solutions de description de services actuels comportent plusieurs défauts:

- ils n'ont pas un langage expressif, des représentations et des outils appropriés pour une si vaste gamme de services;
- les solutions actuelles utilisent une approche basée sur la recherche d'attributs ou noms des services. Ça apporte une mauvaise précision dans les recherches. Les mots utilisés pour la recherche peuvent avoir le même significatif (même sémantique), mais une syntaxe différente. On doit

donc avoir une information sémantique et syntaxique de chaque attribut soit dans la requête soit dans la registration de service;

- le processus de découverte doit supporter aussi des requêtes cessantes ou persistantes, comme aussi la composition de services.

Il a été abordé dans la littérature [4] que pour palier à ses défauts, la description de service doit comprendre le nom des opérations, le type et séquence de tous les paramètres de l'interface. La description doit être compréhensible par les humains ou les machines. Donc chaque attribut de service doit être décrit sur les niveaux syntaxique et sémantique. Les ontologies ont justement été utilisées à cet effet. La description sémantique du service a deux principaux avantages :

- dépendamment de leur fonctionnalité, les services sont classés hiérarchiquement en groupe. Cette information est utilisée pour renvoyer de manière sélective une requête de services aux autres nœuds du réseau sans toutefois surcharger le réseau;
- trouver des services similaires à celui de la requête, c'est à dire même s'ils ont différents nœuds.

Dans [5], D. Chakraborty et A. Joshi proposent d'exploiter la sémantique offerte par DARPA Agent Markup Language [4]. Les services sont décrits avec l'ontologie DAML + OIL [6]. La Figure 2.2 qui suit présente la manière dont ils ont trouvé performant de décrire les services.

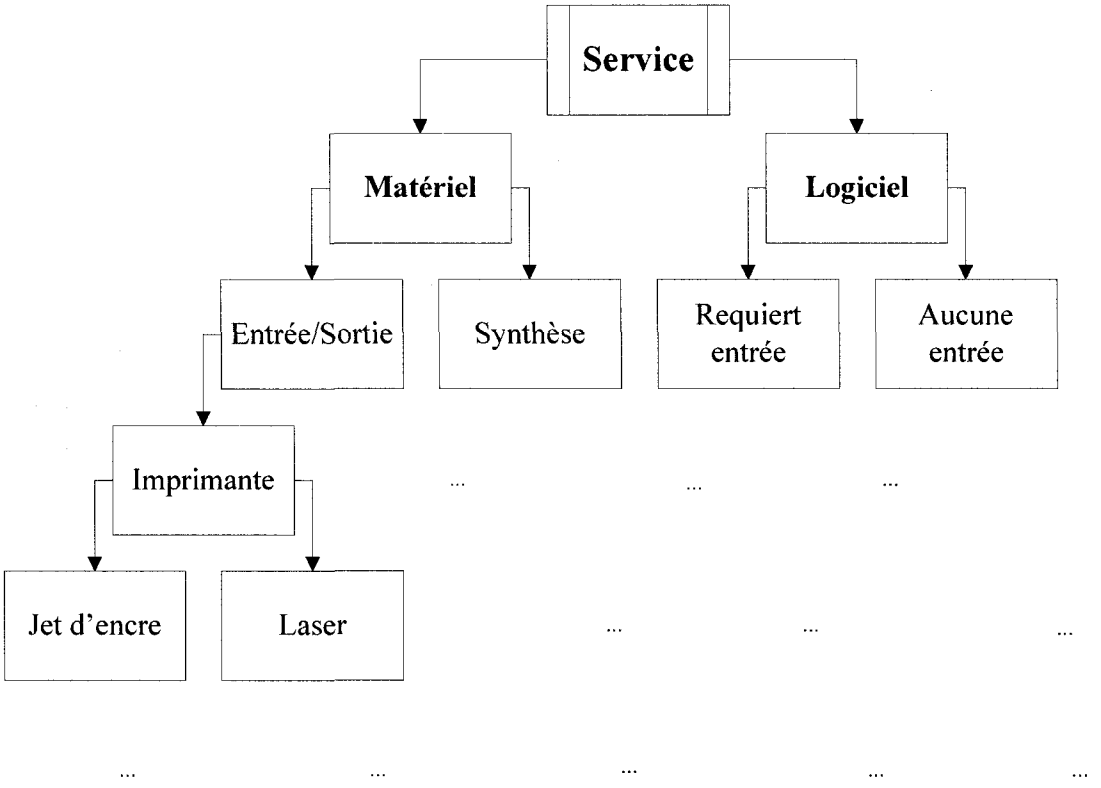


Figure 2.2 Description sémantique des services

Avec cette description, les messages de demande de service et les réponses aux requêtes sont compréhensibles par les deux partis. Cependant, pour être utilisé, un service doit être préalablement découvert par un client (humain ou programme). Les processus de découverte de service dérivent souvent du domaine de l'application. En effet, étant donné le caractère applicatif de certains services, les protocoles de découverte sont, pour la plupart, intégrés aux couches supérieures, soit celle de l'application particulièrement.

2.2.3 Architecture en couche des protocoles

Le routage est un processus de l'architecture qui va servir de point de base pour le déploiement d'une procédure de découverte de service. Quelques approches placent le routage à des couches inférieures à celle de la découverte de service. La Figure 2.3 montre la disposition des protocoles.

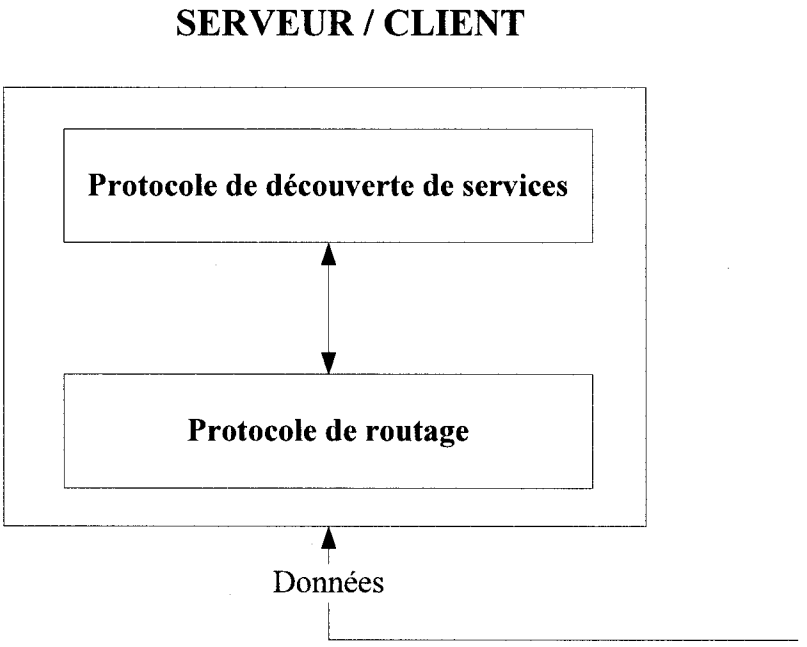


Figure 2.3 Architecture des couches de protocoles

Cette distinction est appropriée pour les services du réseau câblé mais quelques chercheurs critiquent son utilisation dans les réseaux ad hoc. Une étude a été faite [7] pour montrer que l'intégration des deux couches de protocole permet d'augmenter l'efficacité du système. En effet, les auteurs montrent qu'il y aurait plusieurs bénéfices à intégrer la découverte de services au routage :

- utilisation de routes disponibles : le protocole de découverte de services pourra bénéficier des multiples routes qu'un protocole de routage rend possible pour accéder à un serveur donné;

- réduction des entêtes de routage : réutilisation de l'infrastructure de routage pour la découverte de services. Quand un service est découvert, n'importe quel protocole de routage peut être utilisé pour le rejoindre.

L'idée est donc de gérer de manière la plus efficace possible l'énergie et la mémoire disponible au niveau des terminaux en intégrant la découverte de services au routage. Quelques approches de découverte de service utilisent les protocoles de routage DSR ou DSDV. Étant donné que les protocoles de routage fournissent le support de routage des protocoles de découverte de services, il serait intéressant de les présenter dans une section succincte.

2.3 Protocoles de routage dans les réseaux ad hoc

Il s'agit pour nous de donner un bref aperçu des mécanismes mis en œuvre dans les protocoles de routage afin d'en concevoir l'agencement aux protocoles de découverte de services. Dans les réseaux filaires, les protocoles de routage utilisent deux algorithmes. Ces algorithmes sont soient basés sur les états des liens ou sur le vecteur de distance. Pour le routage basé sur l'état des liens, chaque nœud maintient une vision de la topologie complète du réseau. Cette vision se trouve à être mise à jour fréquemment. Pour le routage utilisant le vecteur de distance, pour chaque destination x, chaque noeud i maintient un ensemble de distances $D_{ij}(x)$ où j est l'ensemble des voisins du noeud i.

Les protocoles de routage des réseaux ad hoc sont classifiés en groupes. Dans les protocoles de routage proactif, les routes vers toutes les destinations sont déterminées dès le début et maintenues par un processus de mise à jour périodique. Dans les protocoles réactifs, les routes sont déterminées quand elles sont demandées par la source. Les protocoles hybrides combinent les propriétés des deux premiers groupes.

2.3.1 Protocoles de routage proactif

Dans un protocole de routage proactif, chaque nœud maintient les informations de routage concernant tous les autres nœuds du réseau. Cette information de routage est

maintenue dans des tables de routage. Ces tables sont périodiquement mises à jour lorsque la topologie du réseau change. La différence entre les différents protocoles de routage proactif réside dans la manière dont ces derniers mettent à jour et détectent l'information de routage et garde ce type d'information dans les tables. Un protocole des plus connus de ce groupe est Destination-sequenced distance vector (DSDV). Le routage proposé par DSDV exige que chaque station mobile maintienne une table de routage qui regroupe l'ensemble des nœuds du réseau. Pour chacun des nœuds, elle garde entre autres le nombre de sauts nécessaires pour l'atteindre. Ce protocole est lent et génère un fort trafic de routage.

2.3.2 Protocoles de routage réactif

Les protocoles de routage réactif réduisent les entêtes supplémentaires engendrées par les protocoles proactifs en ne maintenant que de l'information sur les routes actives. C'est à dire que les routes ne sont déterminées et maintenues que pour les nœuds désirant joindre une destination donnée. Les protocoles réactifs sont classifiés en deux catégories : routage par la source et routage « hop by hop ». Pour le routage initié par la source, chaque paquet transporte l'adresse complète entre la source et la destination. Pour le routage « hop by hop », le paquet ne fait que transporter l'adresse de destination et celle du prochain noeud. Les deux protocoles qu'on a jugés pertinent de présenter sont Ad Hoc on-demand distance vector (AODV) et Dynamic source routing (DSR).

- AODV [9]: ce protocole est une amélioration de l'algorithme DSDV. AODV construit les routes par l'emploi d'un cycle de requêtes « route request / route reply ». Il maintient les chemins d'une façon distribuée en gardant une table de routage au niveau de chaque nœud de transit appartenant au chemin recherché. Une entrée de la table de routage contient : l'adresse de la destination, le nœud suivant, la distance en nombre de nœuds, le numéro de séquence destination et le temps

d'expiration de l'entrée de la table. Ce protocole maintient aussi des groupes « multicast ». Il constitue par ce fait des arborescences connectant les différents membres des groupes « multicast ». Les arbres sont composés des membres des groupes et des nœuds nécessaires pour connecter les membres. Le protocole AODV a moins d'entêtes de routage que les protocoles de routage proactifs;

- DSR [8]: ce protocole est basé sur la technique de routage initié par la source. Cette dernière détermine la séquence complète des nœuds à travers lesquels les paquets de données seront envoyés. Les deux opérations de base du protocole sont : la découverte de routes et la maintenance de routes.

2.3.3 Routage prédictif

Chacun des protocoles précédents apporte, à sa façon, de nouvelles optimisations au routage dans les réseaux mobiles ad hoc. S'il existe plusieurs routes disponibles entre une source et une destination, les protocoles sélectionnent celle ayant le moindre nombre de sauts. Une autre approche au routage serait de distinguer la route ayant la durée de vie la plus longue, c'est-à-dire celle maintenant plus de temps la connectivité entre la source et la destination. Le routage prédictif [10] a été proposé pour permettre la sélection de la route ayant la durée de vie la plus longue parmi plusieurs disponibles. Les auteurs E.M Royer et C-K Toh argumentent que le routage prédictif permet une meilleure utilisation de la bande passante en réduisant le taux de signalisation dans le réseau. Le principal argument que les auteurs défendent est qu'en sélectionnant la route dont la durée de vie est la plus longue on retarde l'émission de messages de signalisation pour découvrir une nouvelle route.

Dans le routage prédictif, pour chacune des routes disponibles entre la source et la destination, on prédit le temps d'expiration de chaque lien (LET : Link Expiration Time) composant la route et on calcule le minimum entre tout ces temps. Ce minimum

est le temps d'expiration (RET : Route Expiration Time) supposé pour la route. Ensuite, on sélectionne la route ayant le RET le plus grand pour communiquer avec une destination. Pour déterminer le temps critique *Tc* avant lequel la source doit reconstruire une route pour ne pas perdre la communication en cours: la source retranche du RET le temps *Td* de l'émission du dernier paquet, soit $Tc = RET - Td$.

Pour prédire le LET d'un lien *(i, j)* d'une route, ces protocoles supposent d'abord que chaque nœud soit équipé de GPS qui lui fournit ses paramètres de mouvement (position, vitesse et direction). Ensuite, les paramètres de mouvements des deux nœuds composant le lien sont utilisés dans la formule suivante :

$$LET = \frac{-(ab+cd)+\sqrt{(a^2+c^2)r^2-(ad-bc)^2}}{a^2+c^2} \qquad (2.1)$$

où,

$a = v_i \cos\theta_i - v_j \cos\theta_j$
$b = x_i - x_j$
$c = v_i \sin\theta_i - v_j \sin\theta_j$
$d = y_i - y_j$
(x_i, y_i) les coordonnées d'un émetteur i
(x_j, y_j) les coordonnées d'un récepteur j
v_i et v_j les vitesses des nœuds i et j, respectivement
θ_i et θ_j $(0 \leq \theta_i, \theta_j < 2\pi)$ les vecteurs de mouvement des nœuds i et j, respectivement
Si $v_i = v_j$ et $\theta_i = \theta_j$, alors $a = c = 0$ et LET est infinie

$$RET = \min\{LET_i\}_r \qquad (2.2)$$

où,

r, route disponible entre la source et la destination
i, lien sur une route r

La Figure 2.4 illustre un exemple où deux routes existent entre une source A et une destination F : la première est ABCEF avec 3 comme RET et la deuxième est ABDEF avec 4 comme RET. Par conséquent, F sélectionne la route ABDEF pour la communication avec A. Cette sélection est montrée à la Figure 2.5.

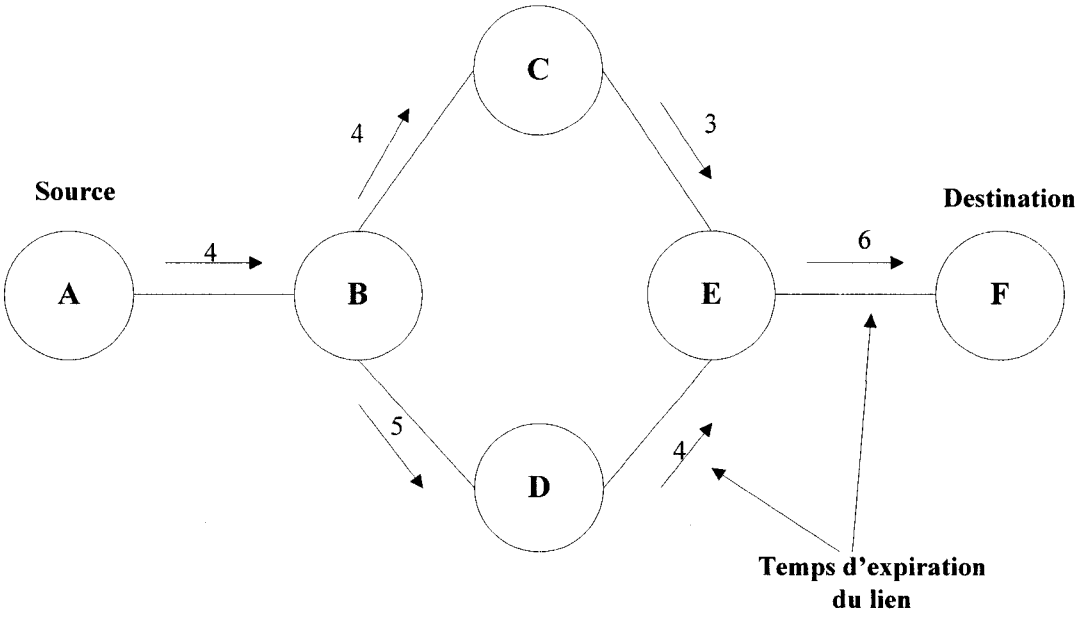


Figure 2.4 Demande d'une route entre les nœuds A et F

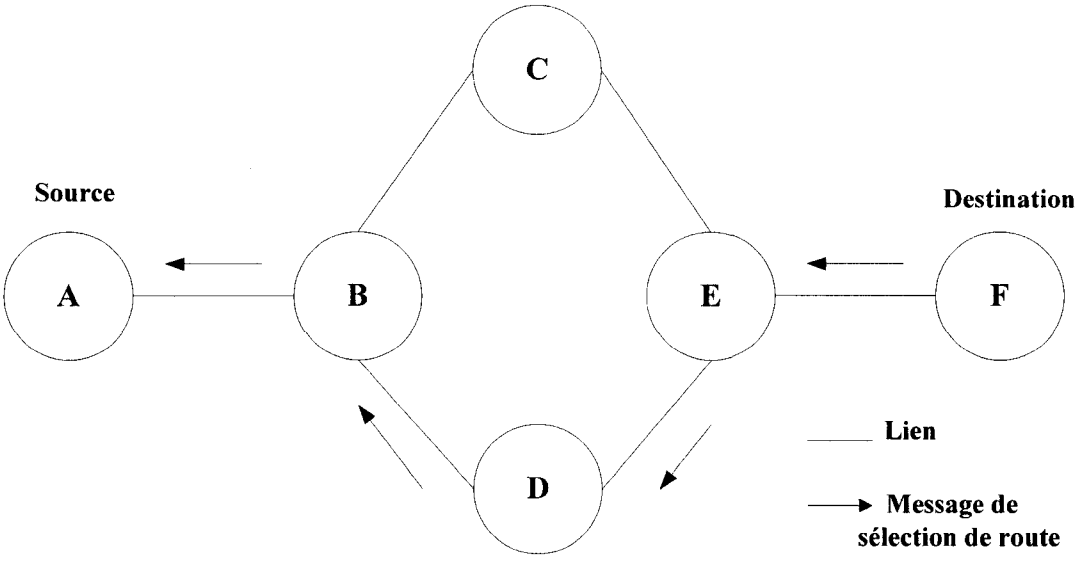


Figure 2.5 Sélection d'une route

2.4 Stratégies de découverte de services

Pour étudier la problématique de découverte de services dans les réseaux ad hoc, on a jugé pertinent d'exposer d'abord les protocoles proposés pour les réseaux filaires, réseaux fixes. Ensuite, on poursuivra avec les approches qui ont été élaborés pour les réseaux mobiles à un saut. Dans une section subséquente seront présentés les protocoles de découverte de plus grande envergure, c'est-à-dire ceux pour les réseaux Manet.

2.4.1 Découverte de services dans les réseaux filaires

Un mécanisme qui tient souvent lieu de référence dans la littérature pour la découverte de services est Jini [11]. C'est une architecture pour les réseaux filaires, elle est distribuée et orientée service. Elle est développée par SUN Microsystems. Basée sur Java, Jini est composée de trois protocoles : découverte, jonction, recherche. Le protocole utilisé pour l'envoi de données est RMI. Le protocole réseau n'est pas indépendant du langage parce qu'il se base sur des mécanismes de sérialisation d'objets Java. Le service de recherche est trouvé par les clients via UDP multicast. Ces derniers

doivent télécharger le service proxy, qui normalement est un RMI stub qui permet la communication avec le serveur. C'est donc le *Jini Lookup Service* ou *JLS* qui permet à un client de rechercher un service. À l'enregistrement, les services préservent le proxy pour le service dans l'espace virtuel de la machine. La flexibilité et la simplicité de Jini sont adaptées au développement de *Web service*s, services accessibles via une interface Web. Le processus de découverte travaille ainsi : imaginez un disque supportant Jini et offrant un service de stockage persistant. Dès que le disque est connecté au réseau, il diffuse une *annonce de présence* en envoyant un paquet multicast sur un port déterminé. Dans l'annonce de présence, sont inclus une adresse IP et un numéro de port où le disque peut être contacté par le service de recherche. Les services de recherche scrutent sur le port déterminé les paquets d'annonce de présence. Lorsqu'un service de recherche reçoit une annonce de présence, il l'ouvre et inspecte le paquet. Le paquet contient les informations qui permettent au service de recherche de déterminer s'il doit ou non contacter l'expéditeur de ce paquet. Si tel est le cas, il contacte directement l'expéditeur en établissant une connexion TCP à l'adresse IP et sur le numéro de port extrait du paquet. En utilisant RMI, le service de recherche envoie à l'initiateur du paquet un objet appelé un *enregistreur de service (service registrar).* L'objectif de cet enregistreur de service est de faciliter la communication future avec le service de recherche. Dans le cas d'un disque dur, le service de recherche attablerait une connexion TCP vers le disque dur et lui enverrait un *enregistreur de service*, grâce auquel le disque dur pourra faire enregistrer son service de stockage persistant par le processus de jonction. L'inconvénient majeur de cette architecture est le manque d'interopérabilité avec d'autres méthodes ou protocoles, puisque Jini requiert Java.

SLP [12], *Service Location Protocol*, est aussi une architecture pour les réseaux filaires. SLP est un standard de l'IETF pour un service de découverte de service décentralisé, léger et extensible. Il y a trois agents en SLP : utilisateur, service et dossier :

- Service Agents (SA) correspond aux intermédiaires qui proposent les services;

- Directory Agents (DA) correspond à ceux qui les répertorient;
- User Agents (UA) correspond aux agents qui utilisent les services.

Les SA diffusent auprès des DA l'existence des services. Lorsqu'un UA recherche un service, il lui suffit de consulter un DA, qui lui indiquera à qui s'adresser. La solution centralisée de l'IETF n'est pas adaptée pour des réseaux à nœuds volatiles. Les mises à jour seront trop fréquentes sur un réseau volatile pour que l'utilisation de SLP soit envisageable. Ce protocole a également été conçu pour fonctionner sans DA, au prix de communications « multicast » pour effectuer des recherches sur le réseau. Les performances de SLP diminuent, et l'utilisation intensive de l'adressage « multicast » n'est guère concevable sur un réseau sans fil.

2.4.2 Découverte de services dans les réseaux mobiles à un saut

SDP [13], *Service Discovery Protocol*, est spécifique pour des dispositifs Bluetooth, un système de connexion à courte portée. Pour faciliter la découverte d'éléments Bluetooth, les opérations de découvertes sont faites entre serveurs et clients. Un élément Bluetooth peut être à la fois client et serveur. Le serveur maintient une liste d'enregistrements de services qui décrit les caractéristiques des services associés avec le serveur. Chaque enregistrement de service contient l'information sur un seul serveur. Un client peut récupérer l'information à partir d'un enregistrement de service maintenu par le serveur SDP en émettant une requête SDP. Si le client, ou une application associée avec le client, décide d'utiliser un service, il doit ouvrir une connexion séparée avec le fournisseur du service afin d'utiliser ce service.

Généralement, un client SDP recherche les services sur la base de caractéristiques de services désirées. Cependant, il est parfois désirable de découvrir quels types de services sont décrits par les enregistrements de services d'un serveur SDP sans aucune information préalable sur les services. La portée de Bluetooth et sa facilité d'utilisation en font un découvreur de services efficace à faible distance. Il ne serait point efficace pour les réseaux de plus grande taille.

2.4.3 Inconvénients des protocoles de découverte de services existants

Il existe plusieurs inconvénients aux protocoles de découverte de services dans le cadre de leur transposition aux réseaux ad hoc. Ainsi, des systèmes tels que Jini offrent le moyen d'annoncer et de découvrir dynamiquement des services, mais exigent de disposer de serveurs référençant les services disponibles. Or, dans les réseaux ad hoc de telles exigences ne peuvent pas nécessairement être satisfaites. Le protocole SLP adopte une approche similaire à celle de Jini en permettant à chaque service ou équipement d'annoncer sa présence au sein du réseau. Cette annonce de présence des services repose sur la diffusion d'un message spécifique. Pour un réseau mobile conçu comme un prolongement d'un réseau d'infrastructure, la centralisation des services applicatifs se révèle être une solution pertinente en termes de facilité de gestion et de découverte des services. En revanche, dans les réseaux mobiles sans infrastructure capables de se former dynamiquement, il n'existe pas d'équipements privilégiés sur lesquels on puisse centraliser les services applicatifs. Considérant qu'il n'est pas judicieux de désigner des équipements particuliers pour jouer le rôle de serveur dans les réseaux Manet, d'autres solutions de découverte de service proposées dans la littérature se sont penchées sur la mise en place de protocoles plus appropriés pour les réseaux mobiles.

2.5 Protocoles de découverte de services pour les réseaux Manet

Les protocoles de découverte de services dépendent du temps. Le temps étant le critère principal de complexité, un service disponible à un instant donné n'est probablement plus accessible à un moment ultérieur. Dans ce qui suit, on présente des protocoles qui tiennent compte de cette caractéristique. De conception extravagante, les protocoles seront critiqués pour comprendre encore mieux les requis d'un protocole de découverte de service pour les réseaux Manet.

2.5.1 Protocole de découverte utilisant des séquences de rondes

Dans [13], les auteurs supposent *(P, Q)* une paire de fonction où *P(s)* est l'annonce de service et *Q(c)* est la requête. Avec *1,2,..., k* où *k* types de services sont offerts dans le réseau. Chaque serveur *s* poste un service de type k_s à un ensemble de nœuds N_s selon l'algorithme *P(s)*. De la même manière, chaque client *c* demande un service k_c d'un ensemble de nœuds N_c selon l'algorithme *Q(c)*. Le client trouve le service si ce dernier a été posté à un nœud que le client demande. Pour s'adapter dynamiquement au changement de topologie, la solution qu'ils ont envisagée est constituée de stratégies. Des stratégies sont effectuées dans une séquence de rondes. À chaque ronde *r*, une paire de *(P(s), Q(c))* est envoyée. La séquence est alors *(P1(s), Q1(c)), (P2(s), Q2(c)), ..., (PR(s), QR(c))*, où *R* est le nombre maximum de rondes avant que la stratégie prenne fin. Pour modéliser la topologie changeante, une probabilité *p* est introduite et indique la qualité de connexion du lien de communication entre une paire de nœuds. Voici les cinq stratégies prisées :

- stratégie 1 (Greedy) : tous les nœuds annoncent et tous les nœuds recherchent tous les nœuds du réseau en utilisant le mécanisme de « broadcast ». Cette stratégie est non adaptive car à chaque ronde l'exécution est la même;
- stratégie 2 (Incremental) : pour utiliser moins de ressources, dans une première ronde, les serveurs et les clients annoncent et recherchent un petit ensemble de nœuds pour ensuite graduellement augmenter le nombre de requêtes et annonces;
- stratégie 3 (Uniform memoryless) : les serveurs annoncent tous les services qu'ils offrent à un ensemble aléatoire de nœuds. Les clients aussi envoient des requêtes de services de manière aléatoire. Le nombre de requêtes affecte le temps d'attente, surcharge le réseau et la probabilité que le client $c \in N$ trouve le service recherché;
- stratégie 4 (With memory) : chaque client construit un « cache » pour stocker les adresses des nœuds auxquels il a déjà posté ou recherché. Chaque nouvelle ronde contacte des nœuds qui ne sont pas dans le « cache » ;

- stratégie 5 (Conservative) : cette stratégie est basée sur la stratégie 1 et requiert qu'à chaque ronde, tous les serveurs (clients) annoncent (recherchent) de leur voisin immédiat (à un saut) en utilisant un mécanisme de « broadcast » à un saut.

Le principal défaut de ces cinq stratégies est que chaque nœud du réseau est soit serveur ou client. Ceci rend l'approche proposée par les auteurs dans [13] moins intéressante pour des réseaux dynamiques où la stabilité d'un ordinateur portatif, par exemple, peut être utilisée et en même temps ce dernier peut vouloir utiliser un service sur le web.

2.5.2 Protocole de couverture de services basé sur l'écoute en local

De la même manière, S. Chien-Chung, S. Thapornphat et R. Liu, Z. Huang [14] proposent un autre protocole de couverture de services basé sur la découverte d'un «dominating set» dans le réseau. Des solutions à ce problème existent mais utilisent une entité centralisée, pour topologie fixe. Ils proposent un protocole distribué avec désynchronisation complète d'élection de médiateurs (nœuds du « dominating set »). Le réseau est ainsi composé de trois types de nœuds, médiateurs, « provider » ou ordinaires. Cette élection est basée sur l'écoute en local (MAC). En effet, chaque nœud « provider » se met en écoute, s'il n'est pas couvert, il se déclare médiateur. Pour augmenter la robustesse du protocole, ils supposent qu'un noeud peut être couvert par plus qu'un médiateur. L'idée proposée est présentée à la Figure 2.6. Dans un réseau ad hoc, un noeud ne peut émettre que si tous ses voisins écoutent, c'est à dire, qu'ils ne sont pas en train d'émettre. Sur ce principe :

- chaque médiateur annonce périodiquement son identité et les nœuds appelés « provider » qu'il couvre;
- chaque nœud « provider » étant en écoute, déclare ses services périodiquement;
- les messages de signalisation sont des messages du protocole;
- chaque nœud « provider » a un lien bidirectionnel avec le médiateur qu'il couvre;

- les liens des médiateurs avec les services qu'ils couvrent ne sont pas forcément bidirectionnels : cette approche tient compte de la présence de liens unidirectionnels.

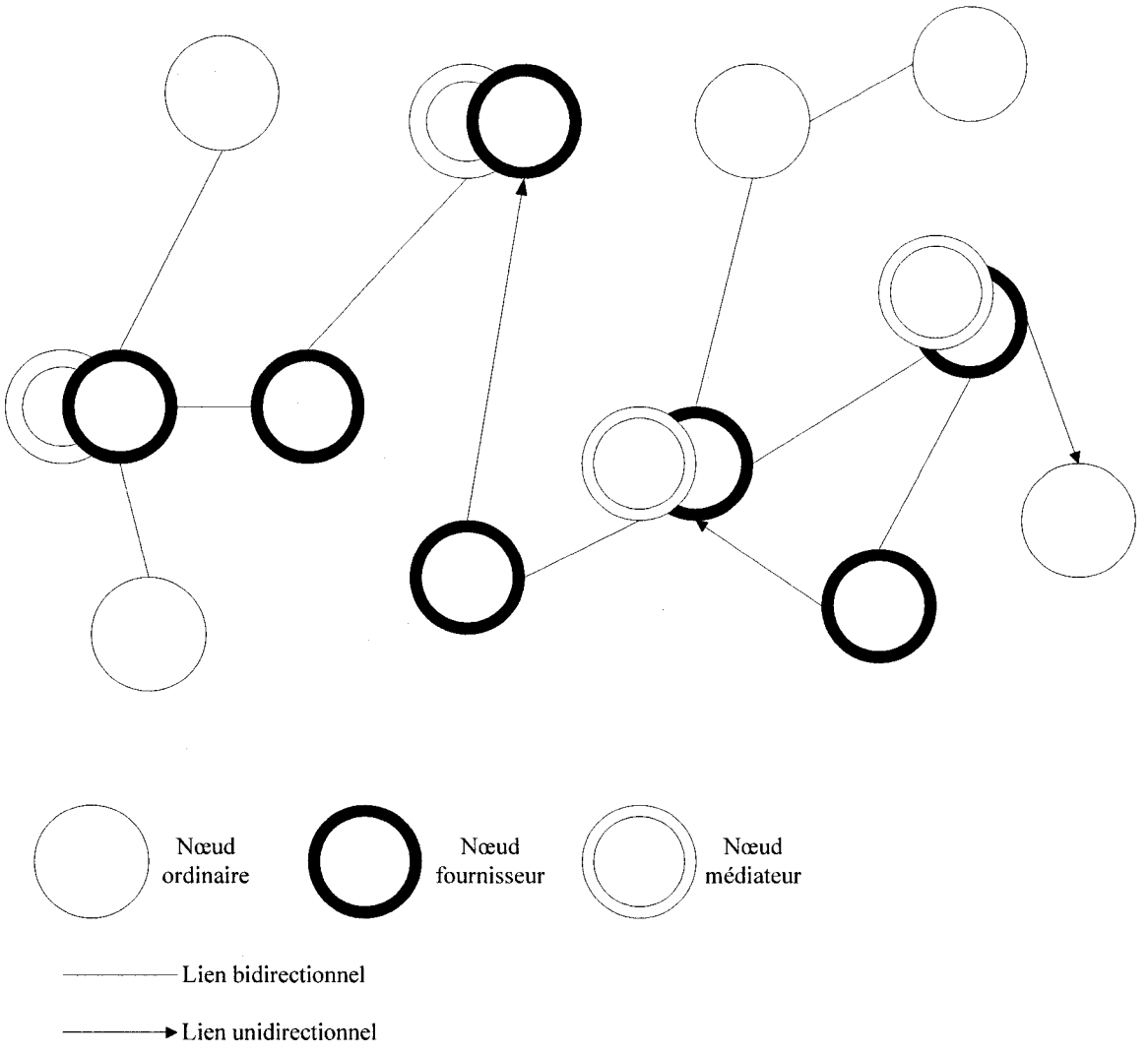


Figure 2.6 Couverture de service basée sur l'écoute en local

D'un autre côté, cette approche doit faire face à des corrections. En effet, de multiples médiateurs peuvent se trouver dans une zone et couvrir différents « provider ». Ainsi, un médiateur englobe les « provider » d'un autre. De plus, si plusieurs «provider»

passent par une zone non couverte, tous les « provider » en question deviennent des médiateurs. Les auteurs ont proposé une reconfiguration pour pallier à ces points faibles. En premier lieu, la durée de vie des médiateurs a été limitée. Ensuite, des conditions ont été mises en place telles que le médiateur doit achever sa session quand sa durée de vie expire ou quand il est couvert par un autre médiateur. De cette manière, ils ont minimisé la redondance et équilibré de la charge. En conclusion, cette approche distribuée basée sur une caractéristique de la couche MAC se porte très mal dans le cas où il existe de la congestion dans le réseau. Dans ce cas, plusieurs collisions surviennent déstabilisant ainsi totalement le réseau.

2.5.3 Découverte de services avec ODMRP

Les auteurs dans [15] proposent un protocole de découverte de service où les informations de signalisation se rapportant à l'annonce et à la découverte du service sont concaténées aux paquets du protocole de routage ODMRP (On-Demand Multicast Routing Protocol). L'architecture exige alors que les nœuds participants au réseau supportent le routage ODMRP. L'approche supporte deux modèles, soient le « Push » et le « Pull » qui sont décrits dans ce qui suit.

- dans l'approche du « Push », l'idée de base de la proposition suggère que si un nœud offre un service, il est alors considéré serveur et il suffit pour lui d'envoyer en « multicast » dans ODMRP une annonce à tous les autres nœuds du réseau. Ainsi, s'il y a des noeuds intéressés par cette annonce, ils n'ont qu'à l'enregistrer dans leur registre local de services. Ils envoient ensuite un paquet de réponse au serveur en question. À ce moment, le serveur répond au nœud par un message « Join », spécifiant l'adresse IP du groupe « multicast » correspondant au service en question;
- dans l'approche du « Pull », l'idée est qu'un service correspond à un numéro d'identification unique. Si un nœud cherche un service, il trouve le numéro d'identification y correspondant et envoie une requête à une adresse « multicast » connue. Cette adresse représente alors le numéro de

groupe « multicast » du service en question. Si un nœud du réseau reçoit la requête et supporte le mécanisme « Pull », il attend un temps aléatoire pour voir s'il n'y aurait pas une réponse dans le réseau à cette requête avec les dernières nouvelles concernant le service. S'il y a eu une réponse, il ignore le message, sinon, il répond avec de l'information qu'il possède sur la demande.

Ainsi, le protocole n'envoie que des messages de mise à jour pour éviter d'envoyer périodiquement les messages « multicast ». De cette manière la signalisation dans le réseau est réduite de façon considérable. Ne reste que le principal inconvénient de la méthode est que seuls les nœuds supportant le routage ODMRP peuvent participer à la découverte de service. Ce qu'on retient c'est que mettre en place un standard qui tienne compte de cette promesse constituerait une limite au design et implémentation des unités mobiles. En effet, dans un environnement ad hoc très diversifiés, les entités ne peuvent être jugées que sur des critères d'admissibilité qui soient souples.

2.5.4 Protocole de découverte et livraison de services, Konark

L'architecture nommée Konark [16] est menue d'un mécanisme décentralisé qui donne la possibilité à chaque nœud du réseau d'annoncer et de découvrir les services dans le milieu. La procédure de description de service est basée sur le langage XML. La description est alors compréhensible à la fois par la machine et par l'homme. L'approche requiert qu'un micro-serveur « http » soit implanté au niveau de chaque nœud pour pouvoir se charger de la livraison des services, livraison basée sur le protocole SOAP. SOAP est un protocole de transmission de messages. Il définit un ensemble de règles pour structurer des messages qui peuvent être utilisés dans de simples transmissions unidirectionnelles, mais il est particulièrement utile pour exécuter des dialogues de type requête / réponse. Les hypothèses que posent les auteurs de Konark sont les suivantes :

- il assume que le réseau supporte une connectivité à la base du Internet Protocol (IP);

- il spécifie que chaque nœud devrait avoir un registre local pour pouvoir stocker les services offerts par la machine.

Konark cherche à supporter, en plus des services classiques d'imprimantes, fax, caméra, audio, des services variés tels que des jeux, music, commerce, météo, etc. Pour ce faire, il définit les services avec le format XML, qui se trouve à être un langage riche permettant de décrire de multitudes de services, un peu comme WSDL. Konark suppose aussi que le réseau supporte le routage « multicast » pour la diffusion des requêtes d'annonce de service. De plus, il propose un registre de services basé sur la méthode d'arborescence. L'approche consiste à ce que chaque nœud du réseau maintienne un «SDP manager» qui sera responsable de la découverte de service et de l'enregistrement des services locaux. Tous les nœuds dans une zone locale joignent ainsi un groupe «multicast». Le «SDP manager» de chaque nœud maintient ainsi un *cache* appelé «Service Registry» pour stocker les services locaux en plus des annonces qu'il reçoit des autres nœuds. Pour découvrir les services dans le réseau, les serveurs emploient le mécanisme de « Push », et les clients utilisent le mécanisme de « Pull », décrient plus haut. En premier lieu, un client initie un message de découverte de service et l'envoie à l'adresse du groupe multicast. À la réception d'un tel message, chaque serveur vérifie la correspondance de ce service avec ceux dans son registre de services. Si un tel service y est, le serveur construit un message d'annonce de service et le retourne au client. Le mécanisme d'annonce de service est similaire à celui de la découverte dans le sens où dès la réception d'un message d'annonce, les clients l'insèrent dans leur registre de services à l'endroit approprié dans l'arbre.

Cette approche essaie d'envoyer les messages «multicast» de façon efficace. Malgré ce fait, il ne reste que pour converger le plus possible vers un protocole de découverte de services qui soit le plus adapté au dynamisme des réseaux Manet, elle requiert une transmission assez importante de messages «multicast» remplissant le réseau de messages de signalisation.

2.5.5 Découverte de service par une approche «multi-cluster»

H. Koubaa et L. Inria Lorraine [17] proposent une organisation dynamique des services en «cluster». Le terme «cluster» signifie regroupement. Plus précisément, ils veulent former des regroupements d'entités basés sur la proximité sémantique et physique. La proximité sémantique désignant le fait que les deux nœuds offrent des services similaires. Pour ce faire, ils proposent qu'une ontologie existe dans le réseau pour décrire les services offerts par les nœuds.

La hiérarchie en «cluster» pour la découverte de service peut être utilisée de la manière suivante : quand un nœud cherche un service, il vérifie si ce dernier n'est pas présent dans son propre niveau de «cluster». Si ceci n'est pas le cas, la requête est transférée à un niveau plus haut dans la hiérarchie de «cluster» jusqu'à l'atteinte du service recherché. À ce niveau, la réponse est renvoyée au bas pour atteindre le niveau du «cluster» qui a initié la requête. Ainsi, au bas de la hiérarchie se trouvent les nœuds offrant des services qui sont similaires en plus d'être physiquement proche les uns des autres. Dans les niveaux plus hauts, les clusters sont formés de terminaux offrants des services dont la description devient de plus en plus générale au fur et à mesure qu'on augmente dans l'arbre de la hiérarchie.

Le plus grand avantage de cette approche est que si un nœud cherche un service en particulier, il sera en mesure de retrouver le plus proche voisin offrant ce service. Un client demandant un service doit avoir comme réponse le serveur qui lui est le plus proche pour ne pas gaspiller de bande passante et surcharger le réseau. De cette manière, on diminue le trafic dans le réseau en offrant des services à deux entités qui sont les plus proches les unes des autres. Cependant, le plus grand inconvénient est qu'il ne peut y avoir plus d'un service par machine, ce qui se trouve à être irréaliste dans les réseaux actuels. De plus, l'ontologie qui est à la base de la proposition doit être connue d'avance par toutes les machines pour pouvoir faire rouler l'algorithme de «cluster» en question.

2.5.6 Protocole de découverte hybride basé sur le groupement

C. Lee et Al. [18] proposent un protocole hybride dans le sens où il combine les deux approches, soit celle basée sur la requête de services et celle sur la diffusion du service. Dans ce protocole hybride suggéré, les nœuds serveurs diffusent les services offerts mais cette diffusion n'est pas faite en «broadcast». Les auteurs définissent un paramètre appelé diamètre qui donne la distance de propagation de ce message. Dans d'autres mots, ce paramètre spécifie le nombre maximum de nœuds que le message de service doit parcourir. Chaque nœud maintient un *cache* pour stocker tous ces messages de services. Puisque les nœuds ne reçoivent pas de messages du réseau entier, ils ne sont pas surchargés en terme d'espace mémoire, batterie, etc. Quand un nœud demande un service, il vérifie d'abord que ce dernier n'est pas dans son *cache*. Si le service voulu n'est pas dans la zone locale du nœud, ce dernier, de la même manière que les serveurs, spécifient un diamètre à la requête et la propage dans le réseau. Si le message atteint un nœud qui offre le service ou un autre ayant dans son *cache* la description d'un serveur offrant le service, le nœud répond à la requête du client en envoyant l'identification du nœud donnant le service. Le principal inconvénient de cette approche est sa robustesse dans le cas où le réseau est de grande taille. On se retrouve vite avec une surcharge du réseau quand ce protocole est utilisé pour les réseaux de grande densité.

2.5.7 Proposition d'un «Backbone» pour la découverte de services

Plusieurs chercheurs s'entendent sur le fait qu'il devrait exister des mécanismes qui différencient les nœuds stables des nœuds instables dans un réseau ad hoc [20]. Étant donné qu'on peut repérer un sous-ensemble de nœuds relativement stables, on peut s'en servir pour former un réseau dorsal, «Backbone». Dans [19], Ulas C. Kozat, G. Kondylis et B. Ryu proposent une architecture de découverte de service décentralisée qui se sert du «Backbone» virtuel pour repérer et enregistrer les services disponibles dans les réseaux Manet.

Ainsi, la proposition se divise en deux étapes : formation du réseau dorsal virtuel et distribution des enregistrements de service, requêtes et réponses. Avec cette proposition, on essaie de repérer en peu de temps le service recherché; les serveurs ne sont pas bombardés. Bref, l'efficacité est améliorée quand le réseau est grand. Ce qu'on doit faire à priori est d'informer tout le réseau des nœuds choisis comme constituant le Backbone. Dans l'algorithme de sélection du Backbone, les auteurs ont tenu compte du critère stabilité, NLFF (Normalized Link Failure Factor) qui représente plus précisément la proportion de pertes sur le lien. Le nœud avec le minimum de proportion de pertes sur le lien se sélectionne comme nœud du «Backbone». Les nœuds se décident en un temps fini et la distance maximale séparant deux nœuds du réseau dorsal est de 2.

La procédure pour la sélection du «Backbone» s'effectue comme suit; d'abord, tous les nœuds sont dits blancs, et seuls ceux avec le plus faible degré de NLFF deviennent noirs. Si après un certain laps de temps un nœud est toujours blanc, il sélectionne un nœud vert, nœud qui est à un saut d'un nœud noir, comme son point d'accès au «Backbone» virtuel. À ce moment, les nœuds noirs formeront le réseau dorsal. En fait, ils encouragent dans cette solution que les nœuds qui restent le plus longtemps dans le réseau deviennent les nœuds du «Backbone», nœuds noirs qui sont stables et se sélectionnent à l'aide du critère de stabilité. L'envoi des messages de contrôle se fait via les messages «Hello» qui se trouvent à être locaux, à un saut et par le même fait très légers pour la transmission.

Après que la première étape du processus aura pris fin, on se trouve en présence d'un «Backbone» virtuel formé des nœuds du réseau dorsal reliés par des liens virtuels. La deuxième étape de l'architecture est d'envoyer efficacement les requêtes et messages d'enregistrement des clients aux serveurs. Les serveurs pourront s'enregistrer auprès des nœuds noirs et les clients demanderont les services qu'ils veulent auprès des nœuds noirs appropriés. Pour ce faire, ils proposent un algorithme en arbre qui sera basé sur la procédure de «multicast» à partir de la source.

L'inconvénient majeur de cette architecture est qu'elle perd son optimalité sous l'effet de la grande mobilité des terminaux. La proposition est lourde dans le sens que

soit un nœud fait parti du réseau dorsal soit il est un voisin de ce dernier. Catégoriser de cette manière les nœuds du réseau contribue une majeure rétribution à l'idée en tant que telle.

2.6 Conclusion

Nous faisons ici l'hypothèse, sinon le constat, que d'après l'état de l'art sur la découverte de service dans les réseaux Manet, les requis d'un protocole de découverte pour ce genre de réseau sont les suivants:

- le nœud ou les nœuds qui auront la responsabilité de maintenir un *cache* ou registre de services, registre contenant la liste des services publiés, doivent avoir une structure arborescente de données pour classifier l'information. De cette manière, la structure facilite le processus de recherche et envoie de services;
- le protocole doit trouver une solution à la situation suivante : au cas où le serveur se déplace ou arrête d'offrir le service à cause de l'énergie limitée, l'architecture doit être en mesure de continuer à offrir le service à un client qui en a fait la requête. La réplication de serveurs pourrait être une solution intéressante dans le cadre d'applications de «web browsing»;
- si un service n'est plus disponible, il faut que le protocole puisse suggérer une alternative, «second best» ;
- l'architecture peut aussi vouloir réduire la congestion vers un nœud serveur. Si plusieurs clients veulent un service qu'un serveur offre, ce dernier va être congestionné. Si on l'exploite indéfiniment, on risque d'épuiser toutes ses ressources. La réplication de service est encore une fois une solution envisageable;
- dans le but de sauver du temps, le protocole doit pouvoir rapprocher les services aux clients mobiles : migration des services. Une solution pourrait profiter d'un réseau dorsal comme celui dans [19] pour déplacer un service ou répliquer le service proche du nœud client. Le serveur

pourra migrer le service d'un client mobile en cours d'exécution vers un nœud stable géographiquement le plus proche. Le client peut continuer à exécuter le service au serveur le plus proche même quand il sort de la région géographiquement couverte par le serveur courant;

Dans ce chapitre, nous avons identifié les propositions comme étant particulièrement adaptés pour adresser les problèmes dans les Manet. Cependant, la décentralisation, la surcharge du réseau et le temps de réponse aux requêtes restent toutefois problématiques dans des environnements aussi étendus et dynamiques que les Manet. Le prochain chapitre présente notre modèle basé sur le déploiement d'un réseau dorsal virtuel pour la découverte et annonce de services.

CHAPITRE III

RÉSEAU DORSAL VIRTUEL POUR LA DÉCOUVERTE DE SERVICES

Dans ce chapitre, nous proposons un modèle pour la découverte de services dans les réseaux ad hoc, en tenant compte des contraintes auxquelles sont soumises les unités mobiles. De façon spécifique, ce modèle utilisera un mécanisme vu dans le précédent chapitre qui différencie les nœuds stables des nœuds instables pour former un sous-ensemble de nœuds constituant le réseau dorsal. Nous commencerons par présenter les réflexions préalables à la conception de notre modèle. Le modèle en tant que tel sera présenté de manière détaillée de même que les algorithmes développés à chaque phase de la proposition.

3.1 Préalable

Tirer le meilleur parti de propositions de protocoles pour résoudre les problèmes associés à la découverte de services dans des environnements ad hoc : telle était l'optique de départ de notre projet. Dans un environnement maîtrisé, la question est vite réglée : l'ensemble des composants logiciels est connu, restreint et les mises à jour sont centralisées. Pour atteindre des objectifs de souplesse et prendre en compte des systèmes à grande échelle, les architectures vues dans le précédent chapitre qui sont basées sur le déploiement d'un réseau dorsal virtuel (RDV) pour repérer et enregistrer les services à l'intérieur d'un réseau dont la topologie change dynamiquement sont apparues très prometteuses. Il s'agissait de former un RDV même dans un état du réseau où la mobilité est très élevée en désignant des unités mobiles clés pour héberger la liste des services offerts dans le réseau. Ainsi, avant d'élaborer, sur cette base, un modèle pour

notre problématique, il convenait de répondre à un certain nombre de questions qui nous ont parus essentielles. Le problème que nous allons tenter de résoudre peut être reformulé de la façon suivante : « *Comment choisir des unités mobiles clés dans un réseau ad hoc pour y stocker des données? Comment augmenter le pourcentage d'accès à l'information tout en réduisant le délai d'accès à celle-ci ?* »

La limitation en bande passante et en énergie oblige à la définition de mécanismes pour réduire la taille des messages échangés. La restriction de l'espace de stockage de l'information du côté des unités mobiles impose la définition de mécanismes pour sauvegarder de l'espace. Plusieurs des protocoles de découverte de services proposés suggèrent que chaque entité dispose d'une mémoire pour le stockage des données auxquelles elle accède fréquemment. La conséquence dans une telle situation est le gaspillage de l'espace. Même si cela garantit des coûts d'accès faibles à la liste de services offerts dans le réseau, et par ce fait, à l'adresse de l'unité offrant le service recherché, il convient de définir un mécanisme qui permette de spécifier les quelques unités mobiles devant disposer d'une mémoire cache pour héberger les données pertinentes à la découverte de service.

La rupture fréquente des liens entre entités d'un réseau ad hoc ainsi que le partitionnement arbitraire du réseau rallonge le délai d'accès à l'information désirée. Il faudra, pour réduire cette latence d'accès, définir une méthode pour minimiser la distance entre le client et l'unité ayant la réponse à la requête de ce dernier. De même, le dynamisme de la topologie du réseau impose la mise en œuvre d'un mécanisme de mise à jour des données liées à la mobilité des entités mobiles.

Pour tenir compte de tous ces requis, le modèle que nous formulons se décortique en trois phases. La première étant le déploiement d'un réseau dorsal virtuel (RDV) le plus stable qui soit, pour repérer et enregistrer les services à l'intérieur d'un réseau dont la topologie change dynamiquement. La deuxième phase est affectée d'un mécanisme de maintenance pour assurer la connectivité du RDV. Et la troisième phase est le processus de découverte et enregistrement de services mis en place pour répondre aux requêtes et annonces publiés.

3.2 Formulation du modèle

On propose le déploiement d'un réseau dorsal virtuel pour repérer et enregistrer les services à l'intérieur d'un réseau dont la topologie change dynamiquement. Une illustration de l'architecture proposée est faite à la Figure 3.1. En effet, on y voit une représentation de 22 unités mobiles constituant le réseau ad hoc. Les quatre unités mobiles, A, B, C et D encerclées en noir représentent les entités spécialisées désignées stables et appartenant au RDV. La délimitation en pointillée de la portée radio d'un nœud stable est dynamique dans le temps. C'est-à-dire qu'une unité mobile quelconque peut acheminer ses requêtes au nœud stable de son choix dépendamment de sa proximité. Les liens virtuels 1, 2, 3 et 4 représentent les échanges entres les nœuds du RDV.

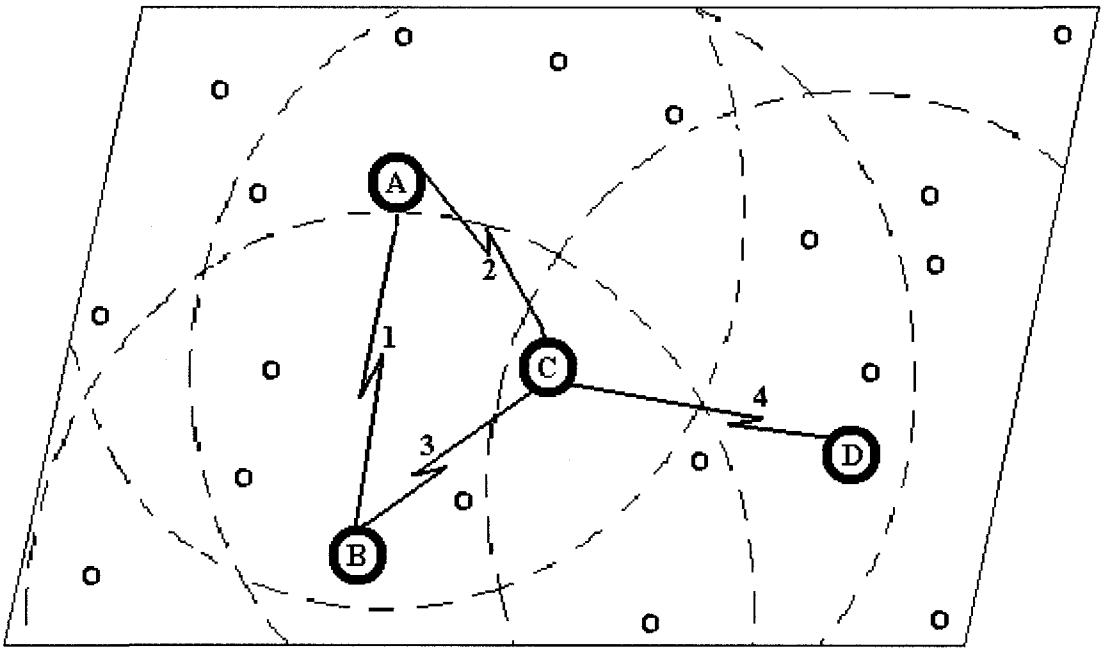


Figure 3.1 Architecture de désignation des nœuds stables du réseau

En effet, les quelques nœuds A, B, C et D qui constituent le RDV sont chargés de maintenir la liste des services offerts dans le réseau. Les autres nœuds du réseau ne maintiennent aucune table et sont parfaitement indépendant de l'architecture mise en

place pour la découverte de service. Contrairement à l'architecture proposée par Ulas C. Kozat et al.,[19], dans le chapitre précédent, qui assume que soit un nœud appartient au RDV ou il lui est voisin, c'est-à-dire que la moitié de la densité du réseau ad hoc est constituée de nœuds stables; Le nombre de nœuds appartenant au RDV dans notre proposition est contingenté. Ainsi, on remarque dans l'illustration que le nombre de nœuds stables est borné, soit quatre nœuds et de beaucoup moindre que la moitié de la densité du réseau ad hoc illustré, soit onze nœuds. Les auteurs de [19] supposent que la moitié des nœuds du réseau sont stables : leurs propos sont contestables. En effet, dans un environnement ad hoc, c'est peu réaliste qu'un nœud sur deux soit stable. Il importe de donner de la valeur au terme stable car à première vue, il n'est point un qualificatif qu'on donnerait aux nœuds d'un réseau ad hoc.

3.2.1 Hypothèses du modèle

Les hypothèses régissant le modèle sont les suivantes : nous considérons un environnement ad hoc avec des nœuds mobiles. Les communications sont établies avec des nœuds se trouvant dans une certaine aire et on suppose le déploiement en deux dimensions des unités mobiles. Tous les nœuds du réseau sont munis d'antennes omnidirectionnelles et sont munis d'une même puissance de transmission. De plus, tous les liens sont bidirectionnels et les terminaux partagent le même canal de communication pour transmettre et recevoir les paquets de contrôle. L'unité mobile est capable de fournir les informations telles que ses coordonnées géographiques, sa vitesse moyenne, l'énergie à disposition de sa batterie ainsi que sa période de résidence dans la portée de transmission de l'émetteur. On suppose aussi que le réseau est connecté, c'est à dire qu'il n'y a pas de partitions dans le réseau. Notre protocole pourra quand même créer un RDV dans chaque sous-réseau de manière indépendante mais ces deux RDV ne seront pas en contact.

3.2.2 Notation et définitions

La notation suivante est utilisée pour la modélisation :

N : Ensemble de tous les nœuds dans le réseau

NIT (Network Information Table) : Table que chaque nœud maintient pour enregistrer des informations sur les nœuds du réseau qu'il peut joindre.

T : Ensemble des nœuds maintenus dans la NIT, $N \subseteq T \not\subset N$

ID_i : Numéro d'identification du nœud i

TTL : (Time To Live) durée de vie d'un paquet (en nombre de saut)

D_i : nombre total de voisins directs du nœud i

BB_i : variable 0-1 telle que : vaut 1 si le nœud i est un nœud du RDV, vaut 0 autrement

S : Ensemble des nœuds appartenant au RDV, soit $N \subseteq T \subseteq S \not\subset T \not\subset N$.

T_E, T_R, T_C, T_w, T_h : Temps d'enregistrement, Temps de reconstruction, Temps de calcul, Temps d'attente, temps message HELLO ou HEY, $T_E > T_R > T_C > T_w > T_h$

Tattenteclient : Temps d'attente d'un paquet ServiceRep

Tattenteserveur : Temps d'attente d'un accusé de réception HelloServeur

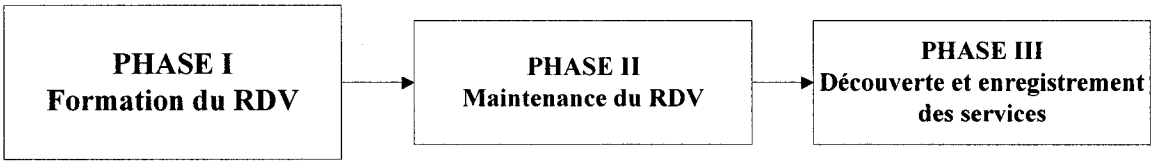
Messages de contrôle : HELLO, HEY, Requête de Service, Message de Publication, Message de Publication Publié, ServiceRep, HelloServeur, JoinRDV

3.3 Architecture générale

Notre première démarche est donc de former le réseau dorsal virtuel qui soit le plus stable possible même dans un état du réseau où la mobilité est très élevée. Cela consistera à désigner des unités mobiles clés pour héberger la liste des services offerts dans le réseau. Nous identifierons ces entités par nœuds du «BackBone» (BB). La désignation judicieuse des nœuds stables requiert la prise en compte de plusieurs paramètres tels que la vitesse moyenne des entités, le nombre total de voisins directs, l'énergie instantanée dont elles disposent, leur capacité de stockage, etc. Par la suite, les nœuds du BB auront la tâche de maintenir la liste des services offerts dans le réseau. Un aspect important de la solution que nous proposons est le mécanisme de maintenance.

Étant donné la mobilité des nœuds, un mécanisme de maintenance doit être mis en place pour assurer la connectivité virtuelle du RDV. Bref, la proposition est sectionnée en trois phases : formation du RDV, maintenance du RDV et découverte et enregistrement des services.

3.4 Phase I : Formation du réseau dorsal virtuel



Il s'agit dans une première phase de sélectionner quelques nœuds du réseau pour former un ensemble dominant d'éléments stables. On procèdera ensuite dans une deuxième phase à la maintenance du réseau dorsal pour tenir compte de la topologie changeante de ce dernier. Plus précisément, il faudra remplacer les nœuds qui quittent le RDV par d'autres qui soient judicieusement choisis et qui devront accomplir les mêmes tâches que ceux qu'ils ont remplacés. La dernière phase consiste à utiliser l'architecture déployée pour enregistrer les services publiés par les serveurs et répondre aux requêtes de services demandées par les clients.

3.4.1 Stabilité d'un nœud

Dans cette section, nous introduisons plus en détail le concept de stabilité d'un nœud. Stabilité et fiabilité sont deux points fortement liés. On peut ainsi qualifier ou décrire un nœud de stable quand on envisage un très haut niveau de fiabilité de la part de ce dernier. L'intégrale des critères tels que l'énergie de la batterie, la vitesse moyenne d'un nœud, l'espace de stockage, le nombre total de voisins directs ainsi que la plus longue période de résidence dans la portée de transmission de l'émetteur donne une

bonne estimation de la stabilité d'un nœud. Il faut, sur la base de ces paramètres, définir l'unité mobile éligible d'être qualifiée de stable. Avant de détailler la procédure de désignation des nœuds stables, nous introduirons quelques terminologies qui serviront à l'élaboration du modèle.

Énergie de la batterie

L'énergie E_j à la disposition d'un nœud à un instant T donné est exprimée par :

$$E_j = E_0 - \frac{1}{T}\sum_{t=1}^{T}(E_t - E_{t-1}) \qquad \textbf{(3.1)}$$

E_o représente l'énergie intrinsèque de l'unité, et la variable E_t les différentes valeurs de l'énergie au cours du temps.

Vitesse moyenne d'un nœud

Chaque nœud calcule sa vitesse moyenne, grandeur donnant une idée de sa mobilité dans le temps de la façon suivante :

$$M_j = \frac{1}{T}\sum_{t=1}^{T}\sqrt{(X_t - X_{t-1})^2 + (Y_t - Y_{t-1})^2 + (Z_t - Z_{t-1})^2} \qquad \textbf{(3.2)}$$

Les variables X_t, Y_t et Z_t représentent les coordonnées des différentes positions occupées par l'unité au cours du temps. *T* représente l'instant auquel cette vitesse est estimée.

Nombre total de voisins directs

Le nombre total de voisins directs d'une unité représente le nombre d'unités dans son voisinage *N(v),* c'est-à-dire l'ensemble des autres nœuds se trouvant dans sa portée radio. Elle est définie de la façon suivante :

$$D_j = |N(j)| = \sum_{j' \in V, j \neq j'} \{dist(j, j') \leq R\} \qquad \textbf{(3.3)}$$

R correspond à la portée radio de l'unité mobile et *V* représente l'ensemble de tous les nœuds environnant l'unité en question.

Période de résidence dans la portée de transmission de l'émetteur

La période de résidence Pij dans la portée de transmission de l'émetteur d'un émetteur i et d'un récepteur j peut être calculée par la formule suivante [20] :

$$Pij = \frac{-(ab+cd)+\sqrt{(a^2+c^2)r^2-(ad-bc)^2}}{a^2+c^2} \qquad \textbf{(3.4)}$$

où,

(x_i, y_i) : les coordonnées d'un émetteur *i*

(x_j, y_j) : les coordonnées d'un récepteur *j*

v_i et v_j : les vitesses des nœuds *i* et *j*, respectivement

θ_i et θ_j : les directions des nœuds *i* et *j*, $(0 \leq \theta_i, \theta_j < 2\pi)$

$a = v_i \cos\theta_i - v_j \cos\theta_j$

$b = x_i - x_j$

$c = v_i \sin\theta_i - v_j \sin\theta_j$

$d = y_i - y_j$

Exceptions : Si $\theta_i = \theta_j$ et $v_i = v_j$, alors $P_{ij} = \infty$

Comme nous l'exposerons plus loin, on pourra tirer des particularités de l'environnement considéré pour arriver à modéliser la procédure de désignation des nœuds stables.

3.4.2 Procédure de désignation des nœuds stables

Il s'agit donc dans une première phase de sélectionner quelques nœuds du réseau pour former un ensemble dominant d'éléments stables. Après avoir énoncé les variables qui permettent de définir l'unité mobile éligible d'être qualifiée de stable : l'énergie de la batterie, la vitesse moyenne, le nombre total de voisins directs et la période de résidence dans la portée de transmission de l'émetteur, on introduit à présent le paramètre LL_i (Link Loss) comme étant le nombre total de liens perdus pour le nœud *i* pour une période de temps d'observation fixe normalisée. Ce paramètre représente en

fait toutes les variables réunies sous forme de combinaison linéaire. Cette pondération des variables est :

$$\begin{aligned} LL_i &= m \bullet E_i + n \bullet M_i + t \bullet D_i + q \bullet P_{ij} \\ m+n+t+q &= 1 \\ m,q,t &\neq 0 \end{aligned} \qquad \textbf{(3.5)}$$

Les variables m, n, t et q sont des facteurs de poids. Ainsi, le paramètre LL_i définit le profil d'éligibilité d'un nœud. L'unité mobile est qualifiée de stable si elle a une faible mobilité dans le temps, possède beaucoup de voisins, dispose de plus d'énergie que la moyenne des autres nœuds et demeure plus longtemps dans la portée de transmission de l'émetteur.

On peut finalement présenter le degré de stabilité d'un nœud comme étant $NLFF_i$ (Normalized link failure frequency), ou fréquence de perte de liens normalisée au niveau de chaque nœud :

$$NLFF_i = \frac{LL_i}{D_i} \qquad \textbf{(3.6)}$$

Pour restreindre le nombre de nœuds parmi lesquels doit se faire la sélection des nœuds les plus stables, on définit $nlff_{th}$ la valeur limite tolérée pour $NLFF_i$.

Degré de stabilité : NLFF

On aperçoit que la valeur de $NLFF_i$ dépend de deux paramètres. Le premier LL_i, correspond au taux de pertes de liens, $0 < LL_i < 1$. Plus cette valeur est petite plus la qualité du lien est meilleure. Le second D_i, correspond au nombre de voisins directs du nœud i. Plus cette valeur est élevée plus grande est la chance de trouver dans le voisinage un nœud qui pourra être utilisé comme substitut de i en cas de perte de lien avec ce dernier. Ainsi, $NLFF_i$ reflète la rapidité à laquelle le voisinage d'un nœud change. Plus cette valeur est faible, plus elle reflète une stabilité. Par exemple, on privilégiera un nœud ayant beaucoup de voisins et une valeur de LL_i quelque peu supérieure à un autre ayant très peu de voisins mais une excellente stabilité de liens. La Figure 3.2 qui suit illustre nos propos :

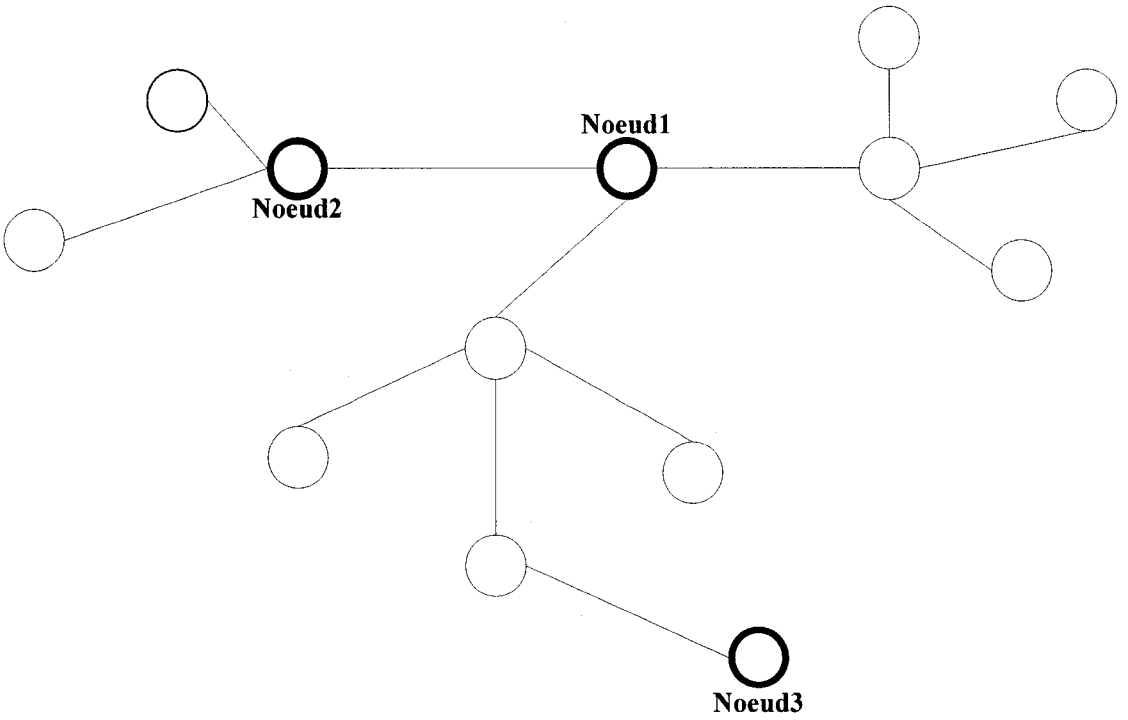


Figure 3.2 Comparaison des degrés de stabilité $NLFF_i$ des nœuds 1, 2 et 3

Voici les cas qu'on a jugés pertinent de présenter :

Cas 1 :

$LL_1 = LL_3 =$ constante

$d_1 = 3$

$d_3 = 1$

- $NLFF_1 = \underline{LL_1} = \underline{cste}$
  $d_1 \quad 3$
- $NLFF_3 = \underline{LL_3} = \underline{cste}$
  $d_3 \quad 1$

$NLFF_1 < NLFF_3$, le nœud 1 est plus stable à cause de son voisinage même s'ils ont le même profil.

Cas 2 :

$d_1 = d_2 =$ constante, mais $LL_1 \neq LL_2$

- $NLFF_1 = \frac{LL_1}{d_1} = \frac{0.9}{\text{cste}}$
- $NLFF_2 = \frac{LL_2}{d_2} = \frac{0.3}{\text{cste}}$

$NLFF_2 < NLFF_1$, le nœud 2 est plus stable à cause qu'il a moins de pertes de liens même s'ils ont le même nombre de voisins.

Cas 3 :

$LL_1 > LL_3$ et $d_1 > d_3$

- $NLFF_1 = \frac{0.7}{3} = 0.0259$
- $NLFF_3 = \frac{0.3}{1} = 0.0333$

$NLFF_1 < NLFF_3$, il est à remarquer dans ce dernier cas que même si le nœud 1 comporte plus de perte de liens, il est considéré plus stable que le nœud 3.

En résumé, si deux nœuds *i* et *j* ont le même nombre total de voisins directs, mais *i* est plus éligible que *j*, alors *i* aura un plus grand NLFF. D'un autre côté, si *i* et *j* ont à peu près le même profil mais *i* a un plus faible nombre de voisins, alors *i* aura encore une fois un plus grand NLFF. Puisqu'une valeur faible de NLFF implique un nœud stable, on prévoit peu de changements dans le voisinage d'un tel nœud. NLFF donne un avantage aux nœuds ayant un voisinage plus abondant.

Dans notre proposition, chaque nœud devra communiquer son degré de stabilité à ses voisins. Dans ce qui suit, nous introduisons plus en détail la procédure mise en place pour former la table d'information qui maintient justement les degrés de stabilité échangés entre nœuds.

3.4.3 Formation de la table d'information, NIT

Initialement, tous les nœuds du réseau ne font pas parti du RDV, alors ils ont une valeur de BB égale à 0. Les nœuds décident ensuite de leur rôle dans le réseau en s'échangeant des messages HELLO. Ces messages de contrôle contiennent tout simplement, en plus du numéro d'identification de chaque nœud, ID_i, une valeur de stabilité lui étant associée, $NLFF_i$ et la table d'information NIT. Le message HELLO est envoyé à chaque T_h secondes par un «broadcast» à 1 saut (TTL=1), et il contient les entêtes suivants :

Type de paquet	NLFF	BB	NIT	TTL

Figure 3.3 Contenu du message HELLO

Ainsi, les nœuds collectent ces messages pour connaître leur voisinage. Chaque nœud forme une table d'information, NIT, contenant la valeur de stabilité associée à chaque nœud dans l'environnement local. Après la réception d'un premier message HELLO, chaque nœud connaît le nombre de voisins immédiats qu'il possède. Il calcule son paramètre NLFF avant d'envoyer un autre HELLO à ses voisins.

L'idée est de dégager au fur et à mesure de l'évolution du système les entités stables du réseau. Les nœuds s'échangent des messages HELLO à chaque T_h secondes et remplissent graduellement de cette manière leur *NIT*, table d'information contenant la liste des nœuds du réseau. Vis-à-vis chaque nœud dans la table se trouve, entre autres, la valeur de $NLFF_i$ correspondant au degré de stabilité du nœud en question.

Dans notre proposition, on prévoit que cet échange de messages HELLO pourra très bien être pris en charge par un algorithme de routage. En effet, les paquets échangés par ce dernier seront légèrement modifiés pour contenir un champ supplémentaire contenant la valeur de NLFF nécessaire à la sélection du nœud le plus stable dans notre proposition.

- À $t = t_1 = T_h = 1$ seconde, après avoir envoyé un premier HELLO de TTL = 1 dans sa surface de transmission, un nœud est certain d'avoir atteint tous ses voisins immédiats et par le même fait, après ce laps de temps $t = t_1$, il aura dans son NIT, une liste ayant au total le nombre et les caractéristiques de ses voisins immédiats.
- À $t = t_1 + t_1$, après échange d'un autre message HELLO contenant la nouvelle NIT, le nœud connaît maintenant les voisins de ses voisins.
- À $t = t_1 + t_1 + t_1$, les nœuds sont en mesure de connaître les voisins des voisins de leurs voisins.

Et ainsi de suite jusqu'à $t = T_w$ correspondant au temps d'attente. À ce moment particulier, les nœuds auront une bonne vision locale de la topologie et seront en mesure de repérer dans leur NIT les nœuds ayant la valeur de NLFF la plus faible. Chaque nœud devra choisir un nombre optimal de nœuds, ces derniers se sélectionnent en fait comme étant les nœuds les plus stables, ils correspondent aux nœuds du RDV.

Cette phase de formation du réseau virtuel de notre proposition est quelque peu similaire à la BMP, Backbone Managment Phase, utilisée dans [19]. Particulièrement dans notre cas, après $T = T_w$ tous les nœuds consultent leur NIT pour d'abord éliminer ceux ayant un NLFF plus élevé que la valeur limite $nlff_{th}$ tolérée car ces derniers sont considérés très instables. Si dans la table du nœud *i*, il n'y a aucun nœud qui a une valeur NLFF < $nlff_{th}$, c'est à dire qu'on est en présence d'un voisinage extrêmement instable, le nœud *i* peut fixer la valeur $nlff_{th}$ à l'infini pour continuer à faire rouler l'algorithme de sélection du nœud le plus stable. Chaque nœud du réseau recherche dans sa table NIT les quelques nœuds ayant les plus faibles valeurs de NLFF. Dépendamment du nombre de voisins qu'il possède dans sa table NIT, chaque nœud choisit un nombre optimal de nœuds stables dans son voisinage. Ces nœuds appartiennent au RDV et ont une valeur de BB = 1.

Élection des nœuds stables

Le degré de stabilité est utilisé pour distinguer les nœuds de la table qui sont les plus stables. Autrement-dit, parmi l'ensemble N des nœuds du réseau, chaque nœud possède dans sa NIT, un sous-ensemble T de N. Il s'agirait à présent de définir la formule permettant de connaître le sous-ensemble S de T, soit $N \subseteq T \subseteq S \not\subset T \not\subset N$. On remarque que cette dernière notation met en lumière la non-inclusivité totale du sous-ensemble S dans T et la non-inclusivité totale de T dans N. Cela précise que seuls quelques nœuds de la NIT appartiennent à S en plus que les nœuds ne possèdent pas dans leur NIT de l'information sur l'ensemble N de tous les nœuds dans le réseau.

$$S = \{\min_i \{T\}\} \qquad (i= 1 , \sum_{j \in T} z_j) \tag{3.7}$$

Avec z_j le nœud du réseau qui appartiennent au RDV et $\sum_{j \in T} z_j$ étant le nombre maximum de nœuds que chaque unité doit choisir dans sa table, il s'agit d'imposer une contrainte permettant de délimiter cette variable. Nous avons déjà précisé que ce nombre ne doit pas être trop élevé pour ne pas avoir trop de messages de signalisation transitant dans le réseau et ni trop faible pour avoir un temps de réponse approprié aux requêtes. Ainsi, la désignation des nœuds stables nécessitent la connaissance du nombre optimal à choisir.

Nombre optimal de nœuds stables

Nous allons, dans cette section, déterminer le nombre optimal de nœuds stables en lui définissant une borne inférieure et supérieure.

$$\begin{aligned} \eta_1 &\leq \sum_{j \in T} z_j \leq \eta_2 \\ \eta_2 &= \frac{T}{2} \\ \eta_1 &= \sqrt{T} \end{aligned} \tag{3.8}$$

Les variables η_1 et η_2 désignent la borne inférieure et supérieure du nombre de nœuds stables qu'il pourrait y avoir dans le réseau en tout temps. Nous avons transposé l'idée

référée par [20] à notre proposition, les auteurs ont estimé la borne inférieure comme étant la racine carrée du nombre d'unités mobiles dans une cellule pour les réseaux VANET. On l'appliquera à notre modèle et on laissera les résultats de performance nous guider vers la borne convenant le mieux au modèle. En plus, on a définit la borne supérieure comme étant celle adoptée par les auteurs de [19], qui stipulent qu'un nœud sur deux dans le réseau est stable. Nous concluons que les résultats de simulation nous donneront le nombre optimal de nœuds stables à choisir, ce nombre se trouvera assurément à l'intérieur des limites η_1 et η_2.

3.4.4 Réseau Dorsal Virtuel

La communication entre les unités du RDV est illustrée à la Figure 3.4.

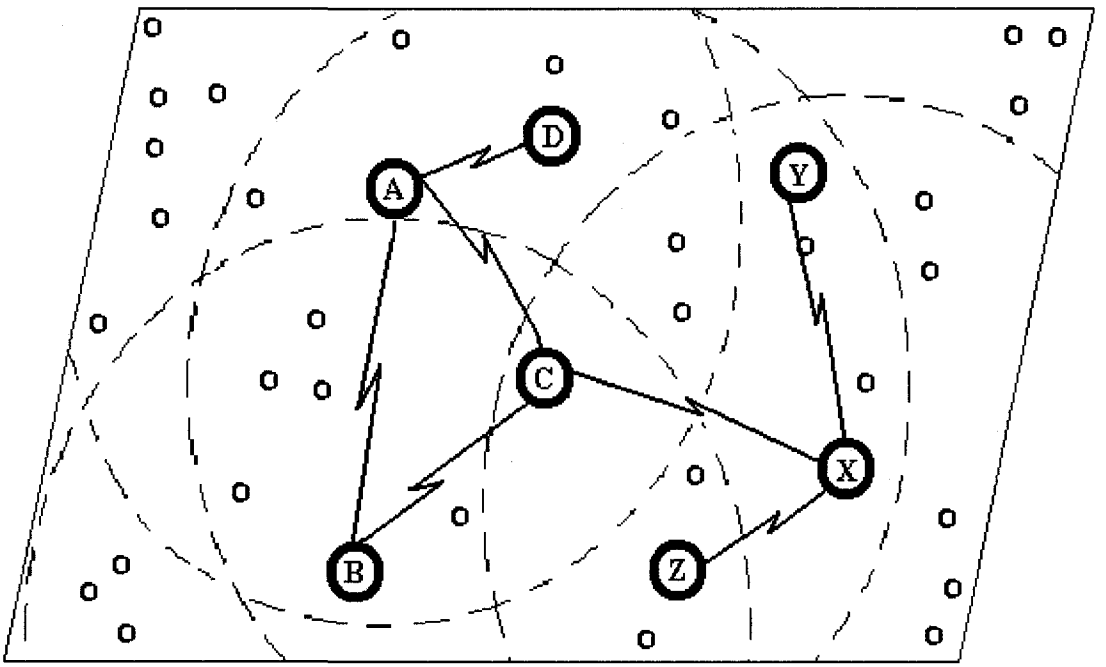


Figure 3.4 RDV, Réseau Dorsal Virtuel

En supposant que le nœud A soit le plus stable dans sa région, ce dernier repère dans sa table NIT d'autres nœuds stables soit B, C et D dans l'exemple, et il communique avec eux à l'aide de messages HEY. Tel que précisé plus haut dans l'équation (3.8), le nombre de nœuds stables que le nœud A choisit dépend de $|T|$, nombre de voisins immédiats dans le réseau et doit faire l'objet d'une analyse particulière qui sera abordée par la suite. Les nœuds stables sont encerclés en noir dans la figure et sont alors considérés $BB = 1$ car ils appartiennent au RDV. De la même manière, le nœud X étant le plus stable dans sa portée repère les nœuds C, Y et Z comme étant d'autres nœuds jugés stables d'après (3.7).

On se retrouve avec un réseau dorsal virtuel constitué de nœuds les plus stables de la topologie. Ainsi, ces nœuds doivent avoir comme champ BB = 1 témoignant de leur rôle dans le réseau. Ils doivent en plus se connaître les uns des autres pour la suite du protocole de découverte de service. C'est le rôle de chaque nœud d'envoyer à ses voisins stables des messages contenant la liste de tous les nœuds que ce dernier a sélectionné comme étant les plus stables. Ce message de contrôle s'appellera JoinRDV et aura comme champs :

Type de paquet	Liste de nœuds stables

Figure 3.5 Contenu du message JoinRDV

À la réception d'un JoinRDV, chaque nœud aura à :

- se considérer BB = 1 pour commencer à assumer ses responsabilités envers le réseau;
- enregistrer dans sa mémoire les adresses des nœuds stables contenus dans la liste qu'il vient de recevoir.

À ce moment, une fois qu'un nœud est étiqueté BB = 1, il doit :

- commencer à envoyer des messages HEY à chacun de ces voisins stables et ceci à intervalle régulier. Cette étape est cruciale au maintient de la stabilité du RDV. La Figure 3.6 qui suit montre le contenu d'un paquet HEY :

Type de paquet	Adresse destination

Figure 3.6 Contenu d'un paquet HEY

- maintenir une table à deux colonnes, tableHEY, afin d'enregistrer les temps de réception des paquets envoyés par leurs voisins.

Si après un certain temps un des voisins d'un nœud stable ne lui a pas envoyé un message HEY, le nœud commence la maintenance pour remplacer le nœud qui est jugé perdu. Cette deuxième phase est expliquée en détail dans la section qui suit. Mais avant de poursuivre, voici en résumé l'algorithme qui permet de générer le réseau dorsal qui soit le plus stable possible.

(A) **Tant que** le temps de la simulation < T_w **faire**

Formuler le paquet HELLO

Envoyer le paquet HELLO aux voisins immédiats

Schéduler le prochain envoie de paquet HELLO

(B) Quand un paquet HELLO arrive

Extraire l'Adresse source du message

Extraire le contenu du paquet

Pour i=1,...|T| de la table NIT **Faire**

Si cette entrée dans la table existe déjà

Mettre à jour la table NIT

Mettre à jour le dataPtr

Sinon créer une entrée dans la table NIT

Insérer contenu du paquet dans la NIT

Fin Si

Si c'est le premier message HELLO des voisins immédiats

Calcul du nombre total de voisins directs du nœud

Calcul de la stabilité NLFF du nœud

Mettre à la table NIT

Fin Si

(C) **Si** le temps de la simulation == T_w

Temp = Contenu de la table NIT_i

Pour k :=1,...|T| de la table NIT_i **Faire**

Retourner min $\{NIT_i\}_k$

Retourner le nombre d'entrées dans NIT_i

Fin Si

(D)Dépendamment du nombre d'entrées dans NIT_i

Calculer $2^{ième}$, $3^{ième}$,$4^{ième}$,... minimum

Mettre à jour la table *TableServeur*

(E)**Pour** j := 1,...|S| de la table *TableServeur* **Faire**

Si l'adresse de l'unité mobile existe dans la table *TableServeur*
    Formuler un paquet de contrôle HEY
    Envoyer le paquet HEY aux autres nœuds dans la table
    Diffuser la liste de nœuds stables
    Charger la table TableHey
    Schéduler le prochain envoie de paquet HEY
    Vider la table *TableServeur*
    Retourner dans l'état SERVEUR
Sinon Vider la table *TableServeur*
    Retourner dans l'état CLIENT
Fin Si

Figure 3.7 Algorithme général de formation du réseau dorsal virtuel, RDV

3.5 Phase II : Maintenance du réseau dorsal virtuel

Après avoir formé l'ensemble dominant qui constitue le réseau dorsal virtuel, on procède à l'étape de la maintenance. En effet, la maintenance est locale de manière à réagir rapidement aux changements dans la topologie. De cette manière, l'algorithme proposé sera décentralisé. Ainsi, si pour une raison ou pour une autre un nœud quitte le réseau dorsal soit par une déconnexion causée par un manque d'énergie ou à cause de son dynamisme, une reconstruction du chemin doit se faire localement. Étant donné que les nœuds s'envoient périodiquement des messages HEY pour s'assurer de toujours être

en contact, quand un nœud ne reçoit rien dans l'intervalle de temps T_h, temps séparant deux messages HEY, il suffit pour lui d'attendre pendant un petit temps supplémentaire T_R, temps de reconstruction, avant de commencer le processus de reconstruction. Le nœud en question règle la situation localement en trouvant un autre nœud stable pour remplacer le nœud perdu. Ce processus est démontré à la Figure 3.8 qui suit.

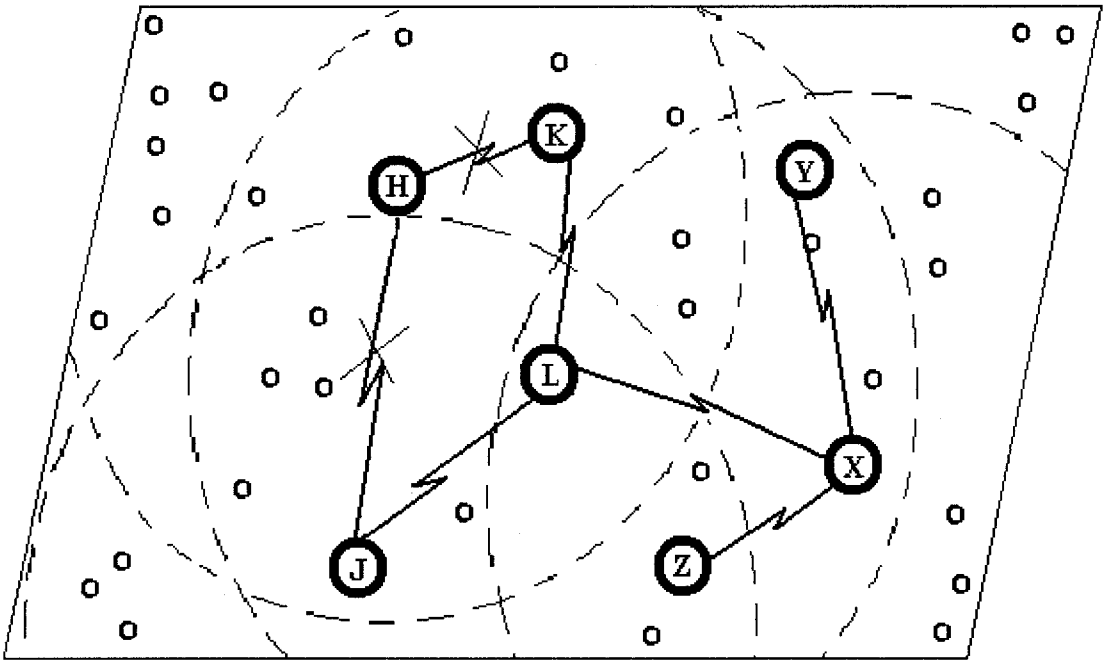


Figure 3.8 Maintenance du RDV

Le nœud H appartient au RDV. Si le nœud H se trouve à être déconnecté du réseau, les autres nœuds du RDV vont réaliser cette situation après $T = T_h + T_R$. En effet, ils n'auront pas reçu le message HEY de leur voisin H qui a quitté le réseau. Après que le nombre de nœuds stables au niveau de chaque nœud aura atteint la limite inférieure donnée par la contrainte η_l, le nœud en question remplacera les nœuds perdus par d'autres nœuds du réseau. Du point de vue de J et K, ils n'ont pas reçu de messages de la part de H en $T_h + T_R$ secondes, alors que du point de vue de H, il n'a rien reçu ni de J ni

de K en $T_h + T_R$ secondes. Pour cette raison, J et K commencent la reconstruction alors que H se prune, c'est-à-dire met à jour son champs BB pour être égal à 0. Ainsi, si un nœud sort du RDV, il faut pouvoir :

1) le pruner en modifiant le champ BB pour qu'il soit égal à 0;
2) le remplacer, si nécessaire, par un autre.

En résumé, tous les nœuds du RDV doivent être en mesure de :

- S'envoyer à intervalle T_h secondes des messages HEY
- Reconnaître que si après $T_h + T_R$ ils n'ont pas reçu de messages HEY, ils n'appartiennent plus au RDV.
- Reconnaître que si après $T_h + T_R$ ils n'ont rien reçu d'un seul nœud, ils doivent le pruner et commencer la maintenance si nécessaire

3.5.1 Maintenance partielle

L'objectif est donc de ne pas surcharger le réseau avec des messages de routage. Après qu'un problème de rupture de lien soit détecté, le nœud qui a détecté l'anomalie se charge lui-même de la réparation, c'est le principe de maintenance partielle. Cette procédure de reconstruction présente l'intérêt d'être rapide et de consommer peu de bande passante. On limite l'aire de prospection et donc le nombre de réémissions des paquets de recherche de nœuds stables. Dans notre proposition, on cherche à former un RDV qui soit constitué de nœuds les plus stables possibles; Par ce fait, on diminue les risques de défaillance de liens car le RDV est supposé fiable. N'empêche qu'étant donné le caractère ad hoc de la topologie, on prévoie un mécanisme de maintenance. Cette maintenance est réalisable car les anomalies sont facilement réparables. Une anomalie est facilement réparable lorsqu'elle peut être réparée par une maintenance locale. La mise en œuvre de maintenance locale aux probabilités de succès rehaussées est assurée par la prise en compte du paramètre NLFF qui témoigne de la stabilité des nœuds dans le réseau.

La procédure de reconstruction est la suivante : si le seuil du nombre de nœuds stables est atteint, les nœuds dans l'exemple de la Figure 3.8 plus haut doivent reconstruire leur NIT pour découvrir quels sont les nouveaux nœuds les plus stables dans leur voisinage. Le nœud J par exemple demande de connaître ses voisins via des messages HELLO, afin de connaître le degré de stabilité, NLFF de chaque voisin. Le nœud K fait pareillement. Quand le nœud J aura rempli sa table d'information NIT, il choisit les nœuds les plus stables, le nœud K aussi. Les nœuds s'envoient des messages HEY pour être informés de leur rôle dans le réseau soit, BB = 1.

Une rupture de lien peut être détectée par le nœud soit en amont soit en aval du lien rompu. Des informations sur la rupture peuvent être fournies par des protocoles de couches inférieures. Par exemple, le protocole IEEE802.11 de la couche liaison peut prendre en charge cette opération. Ainsi, dans le cas où le lien vers le nœud suivant est rompu, dépendamment du seuil, le nœud initie la phase de maintenance. En effet, ceci doit être rare car le paramètre NLFF devrait donner une bonne idée de l'environnement de chaque nœud du RDV, NLFF dépendant du nombre de voisins immédiats. Il ne devrait donc pas y avoir de problèmes pour trouver au moins un nœud dans le voisinage du nœud qui quitte le RDV pour le remplacer.

Après la maintenance, on peut avoir introduit un seul nouveau nœud, soit L comme dans l'exemple de la figure 3.8, ou plus si nécessaire. Les nœuds ainsi introduits devront être informés qu'ils appartiennent au RDV pour qu'ils puissent assumer leurs responsabilités en tant que nœud BB = 1 dans le RDV. De plus, ils devront être informés de la présence des autres nœuds stables du RDV. Ainsi, comme J et K ont la liste de ces nœuds, ils pourront la transmettre à l'aide des messages de contrôle JoinRDV illustrés à la Figure 3.5. Une fois cette dernière étape complétée, peu importe quel nœud quitte le RDV, il sera localement remplacé par un autre qui soit aussi stable. Cette situation représente bien une maintenance partielle dans le sens que la reconstruction est locale. L'algorithme de maintenance proposé est présenté à la Figure 3.9 :

(A) À la réception d'un message HEY

Pour k :=1,...|S| liste de serveurs, **Faire**

Si l'adresse du destinataire n'est pas dans la liste

Insérer adresse dans la *tableHey*

Sinon l'adresse est dans la liste

Inscrire nouveau temps de réception, *lastHeard*, dans la *tableHey*

Fin Si

(B) À la réception d'un message CheckTable

Pour k :=1,...|S| liste de serveurs, **Faire**

Si le temps de simulation – *tableHey*->*lastHeard* $<T_h+T_R$

Effacer ce serveur de la table

Effacer ce serveur de la liste de serveurs

Fin Si

Calculer la borne inférieure dans l'équation (3.8)

Si 0 < borne < borne inférieure

Commencer la maintenance

Envoyer messages HELLO

Enregistrer nouveau nœud stable

Sinon

Si borne == 0

Retourner dans l'état CLIENT

Fin Si

Sinon

Si borne > borne inférieure

Schéduler le prochain message CheckTable

Retourner dans l'état SERVEUR

Fin Si

Fin Si

Figure 3.9 Algorithme de maintenance du RDV

3.6 Phase III : Enregistrement et découverte de services

Maintenant que le réseau dorsal est mis en place, il s'agit d'avoir des mécanismes pour permettre aux serveurs d'enregistrer leurs services et aux clients de prendre connaissance de l'emplacement des services qu'ils désirent.

3.6.1 Publication d'un service par le serveur

Les nœuds du réseau dorsal sont des nœuds stables qui doivent avoir en tout temps une liste des services offerts dans le réseau. Ils s'échangent périodiquement des mises à jour des services qu'ils reçoivent. En effet, quand un serveur désire faire connaître le service qu'il offre, il doit d'abord s'enregistrer auprès de n'importe quel nœud du RDV. Ce nœud aura à son tour la tâche d'informer ses voisins dans le RDV, via unicast, de l'enregistrement qu'il vient de recevoir. De temps à autre, chaque serveur doit renouveler son enregistrement. Ainsi T_E, le temps d'enregistrement correspond au temps que doit attendre un serveur avant de renouveler sa publication. À la réception de ce nouveau message, un nœud du RDV peut tout simplement s'assurer que ce serveur est bel et bien dans sa liste, ceci voudra dire que le service est toujours offert, aucune mise à jour auprès des voisins n'est requise. Si après un temps T_E aucun message de renouvellement n'est reçu pour un service donné, le service se trouve à être enlevé de la liste des services offerts. La Figure 3.10 montre la manière dont un serveur s'enregistre auprès du réseau.

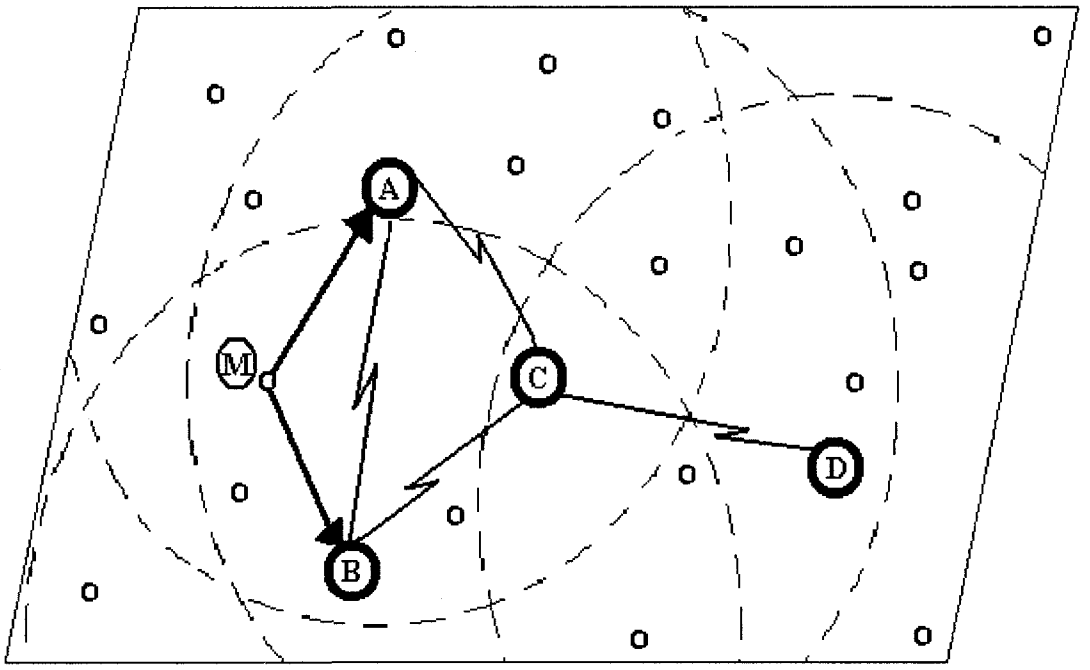


Figure 3.10 Enregistrement d'un service

Le nœud M désire informer le réseau dorsal du service qu'il offre. Il n'a qu'à envoyer un Message de Publication au nœud BB = 1 qui lui est le plus proche avec un nombre de sauts fixe. De cette manière, n'importe quel nœud du réseau, peu importe son emplacement, pourra croiser un nœud du réseau dorsal et s'enregistrer auprès de lui. Le nombre de sauts est plus ou moins faible et dépend du voisinage des nœuds. Il permet de diminuer les messages de signalisation qui transitent dans le réseau en ne dirigeant que localement l'information pertinente. Avec quelques sauts, le nœud M trouve les nœuds A et B du réseau dorsal. À ce niveau, par simple routage unicast le restant des nœuds du RDV seront informés du nouveau service en question. Voici le contenu d'un Message de Publication :

Type de paquet	ID	Adresse source	Taille de la description du service
Description de service	Groupe de service	Nombre de sauts	Durée de vie

Figure 3.11 Contenu d'un Message de Publication

Il est à noter que la taille de la description de service permet d'avoir une liste de services dans le cas où le serveur en offre différents types. Le nœud du RDV qui reçoit ce message édite son ServiceCache qui contient les champs suivants :

ID	Adresse Source	Description du service	Groupe de service	Durée de vie

Figure 3.12 Contenu du ServiceCache

Après la réception du Message de Publication et une mise à jour du ServiceCache, le nœud du RDV envoie d'abord un message HelloServeur au serveur avec comme adresse de destination l'adresse source du message de publication. Voici le contenu du message HelloServeur :

Type de paquet	Adresse Destination

Figure 3.13 Contenu de HelloServeur

Par la suite, le nœud du RDV doit informer le restant des nœuds du RDV du service qui vient d'être disponible dans le réseau. Pour ne pas avoir de livraisons dupliquées ou de cycles infinis dans le RDV, il faut que chaque nœud BB=1 qui reçoit un message de ses voisins vérifie :

- si ID du message reçu est le même qu'une des entrées du ServiceCache, il jette le message. Ceci veut dire qu'il a déjà reçu cette annonce de service;
- si ID est nouveau et que le type de paquet est un Message de Publication, il l'enregistre dans son ServiceCache et envoie aux restants des nœuds du RDV l'enregistrement. C'est que ce message vient d'un nœud serveur en dehors du RDV. Chacun des nœuds stables recevra un message de type Message de Publication Publié dont le contenu est le suivant:

Type de paquet	ID	Adresse source
Description de service	Durée de vie	Groupe de service
Taille de la description du service	Adresse Destination	Adresse nœud offrant service

Figure 3.14 Contenu du Message de Publication Publié

  - Type de paquet : message de publication publié
  - ID : même numéro que l'identification ID du message de publication reçu
  - Adresse source : adresse du nœud en question
  - Adresse destination : adresse d'un voisin immédiat dans le RDV
  - Adresse nœud offrant service : adresse source qui se trouve dans le message de publication reçu
  - Taille de la description de service, Description de service, groupe de service et durée de vie : même champs que ceux du message de publication reçu
- si ID est nouveau et que le type de paquet est un message de publication publié, le message ne vient pas directement d'un serveur mais d'un de ses deux voisins

dans le RDV, le nœud en question enregistre ce service et renvoie ce message de publication publié à son prochain voisin dans le RDV et non celui qui le lui a envoyé. Il ne fait que modifier les champs adresse source en mettant sa propre adresse et adresse destination celle du prochain voisin.

Voici donc le flux de messages pour un serveur désirant annoncer ces services, il est visualisé dans le diagramme de séquence de la Figure 3.15 qui suit :

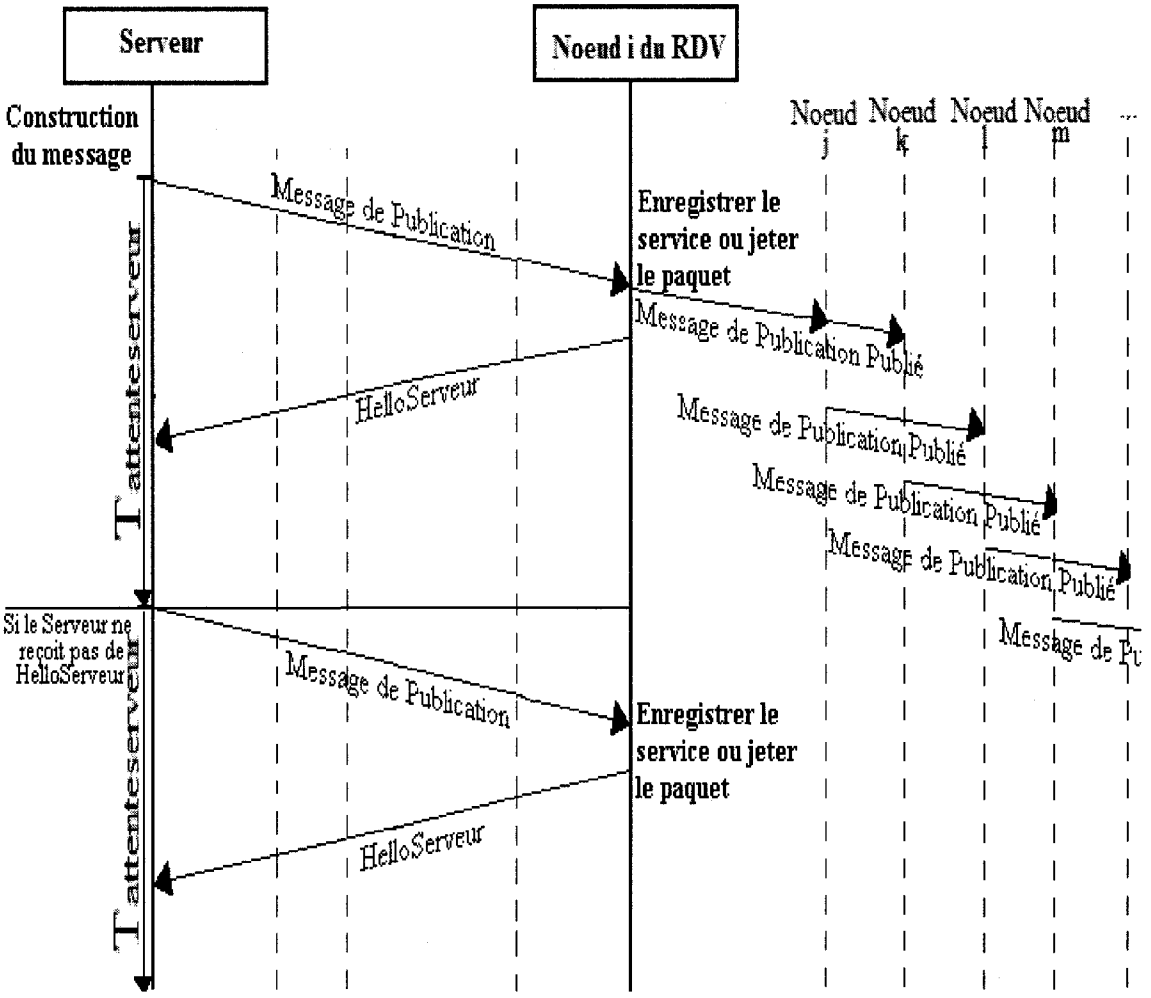


Figure 3.15 Diagramme de séquence pour un serveur

Le serveur formule un Message de publication et part l'horloge à Tattenteserveur :

1) le message transite d'un nœud à un autre dans le réseau pour le nombre fixe de sauts;
2) si un nœud *i* reçoit le Message de Publication et qu'il fait parti du RDV, il effectue des vérifications avec le numéro ID pour s'assurer que le service n'est pas déjà enregistré. Si oui, il l'enregistre sinon, il le jette. Dans les deux cas, il envoie en unicast un message HelloServeur au serveur;
3) dans le cas où le numéro ID du Message de Publication est nouveau, le nœud *i* envoie en unicast un Message de Publication Publié à ses voisins du RDV, *j*, *k*, *l* et *m*;
4) les nœuds *j* et *k*, voisins immédiats de *i* reçoivent le Message de Publication Publié et font la même vérification à savoir si le numéro ID n'est pas déjà dans leur ServiceCache. S'il est nouveau, les nœuds *j* et *k* enregistrent le service et envoient en unicast à leurs voisins immédiats mais non le voisin *i* qui leur a envoyé le message;
5) pareillement, les nœuds *l* et *m* effectuent les vérifications. Et ainsi de suite jusqu'à ce que tous les nœuds du RDV reçoivent le Message de Publication Publié.
6) si après un Tattenteserveur, le serveur n'a toujours pas reçu de message HelloServeur, c'est que soit :
   - le nombre de sauts dans le message de publication est trop faible, il n'a pu atteindre un nœud du RDV. Cette probabilité de ne pas atteindre un nœud du RDV dans un nombre de sauts fixe est très faible parce que c'est la richesse même de notre proposition;
   - le message de publication a été perdu;
   - le message HelloServeur a été perdu.

Quelle que soit la raison, le serveur envoie à nouveau le Message de Publication mais cette fois en doublant la valeur du nombre de sauts, et il repart l'horloge à

Tattenteserveur. On s'assure ainsi que le service sera connu d'au moins un nœud du RDV.

Le point fort de notre proposition est qu'elle fait en sorte que la mobilité du serveur dans le réseau est transparente, c'est à dire peut importe où le serveur va se déplacer dans la topologie, plusieurs nœuds stables (nœuds du RDV) pourront donner son adresse à un client requérant son service. Le client communiquera avec le serveur en unicast via n'importe quelle route fournit par n'importe quel protocole de routage.

3.6.2 Requête de service du client

De la même manière, lorsqu'un nœud désire découvrir un service et qu'il ne fait pas parti du réseau dorsal, il n'a qu'à envoyer une Requête de Service avec un nombre de sauts fixe. Le nœud du RDV le plus proche lui enverra la réponse à sa requête avec un ServiceRep. Encore une fois, la requête a une portée relativement faible et ne correspond aucunement pas à une procédure de «flooding». De cette manière, le trafic dans le réseau est diminué de façon considérable. Voici le contenu d'une Requête de Service :

Type de paquet	ID	Adresse source
Description de service	Nombre de sauts	Groupe de service
Taille de la description du service		

Figure 3.16 Contenu d'une Requête de service

Il est à noter que contrairement au Message de Publication, la Requête de Service ne contient pas de champs de durée de vie. À la réception d'une Requête de Service, le nœud du RDV renvoie un ServiceRep contenant l'adresse du nœud offrant le service demandé:

Type de paquet	Adresse Destination	Adresse nœud offrant service

Figure 3.17 Contenu d'un message ServiceRep

Voici donc le flux de messages pour un client désirant connaître un service, il est visualisé dans le diagramme de séquence de la Figure 3.18:

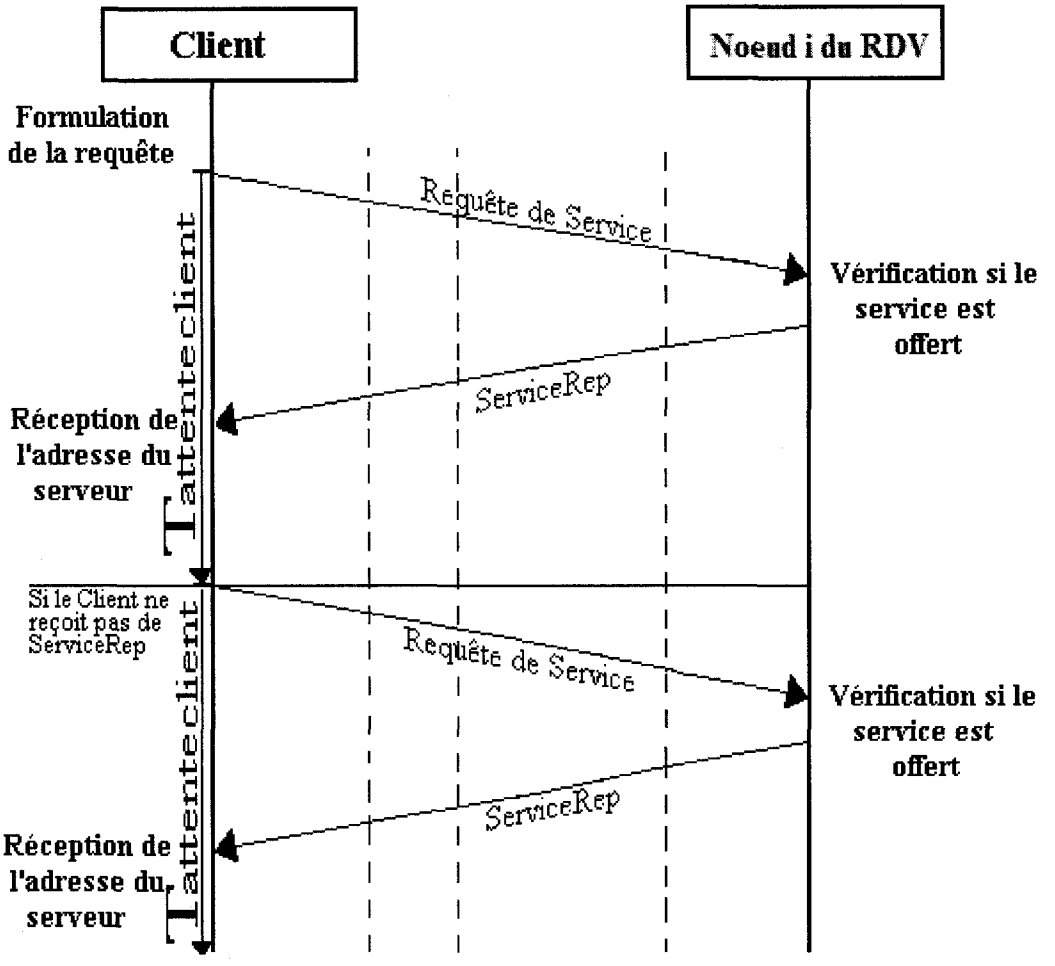


Figure 3.18 Diagramme de séquence pour un client

Le client formule sa Requête de Service et part l'horloge à Tattenteclient :

1) la requête transite d'un nœud à un autre dans le réseau pour le nombre fixe de sauts;
2) si un nœud *i* reçoit la Requête de Service et qu'il fait parti du RDV, il vérifie si le service est disponible dans le réseau. Selon le cas, le nœud *i* envoie en unicast un ServiceRep avec dans le champ 'adresse offrant service' vide si aucun serveur n'offre le service ou rempli avec l'adresse du serveur approprié;
3) si après un Tattenteclient, le client n'a toujours pas reçu de ServiceRep, c'est que soit :
    - la prédiction initiale de la valeur du nombre de sauts dans la Requête de Service est trop faible, le message n'a pu atteindre un nœud du RDV. Encore une fois, cette probabilité de ne pas atteindre un nœud du RDV dans un nombre de sauts fixe est très faible parce que l'architecture proposée mise sur cette caractéristique, nombre de sauts fixes, pour vanter l'efficacité du protocole;
    - la Requête de Service a été perdu;
    - Le ServiceRep a été perdu.

Quelle que soit la raison, le client envoie à nouveau la requête mais cette fois en doublant la valeur du nombre de sauts, et il repart l'horloge à Tattenteclient.

CHAPITRE IV

IMPLÉMENTATION ET RÉSULTATS

Après une présentation de l'analyse théorique du comportement que l'on peut attendre de notre proposition, nous avons entrepris de développer un environnement de simulation suffisamment complet qui nous permette de tester notre approche. La simulation permettra de confirmer aussi bien les résultats de l'analyse théorique que le comportement des algorithmes observés à travers une émulation temps réel du réseau et une mise en œuvre sur un réseau sans fil IEEE 802.11. Dans un premier temps, nous présenterons cet environnement d'implémentation et nous procéderons par la suite à l'évaluation de performance du modèle proposé pour faire notre preuve de concept. Outre la présentation des résultats de simulations obtenus, nous effectuerons une analyse de ces derniers et situerons les performances du modèle par rapport aux performances d'autres solutions proposées dans la littérature.

4.1 Implémentation

La plate-forme utilisée pour la mise en œuvre de notre solution est du type *PC*. C'est une machine munie d'un processeur *Intel Pentium* de 1.80 GHz, avec un système d'exploitation *Windows XP*. L'éditeur *Visual C++ 6.0* est utilisé pour la programmation et la compilation s'effectue avec l'engin *nmake* faisant partie de la suite *Visual Studio .NET (2003)de Microsoft.* Bien que l'environnement OPNET permette la modélisation et la simulation de réseaux de communication grâce à ses bibliothèques de modèles et de protocoles, cet éditeur majeur de solutions d'analyse et d'optimisation de réseaux et d'applications n'est pas l'idéal pour simuler les réseaux WLAN. L'implémentation avec le simulateur Qualnet [21] quant à elle, serait une bonne alternative car elle modélise les couches supérieures et inférieures de manière efficace.

4.1.1 Qualnet

Qualnet, développé par SNT est une suite logicielle dédiée à la simulation de réseaux filaires ou sans fil de tous types (Wifi, GSM, liaisons tactiques, ...). Il se base sur un langage de simulation parallèle dénommé PARSEC (*PARallel Simulation Environment forComplex system*). Ce langage d'émulation est à évènement discret très similaire au langage C. La mise en place d'un protocole s'effectue à l'aide d'une machine d'états dont les transitions s'effectuent à l'occurrence d'évènements; un évènement est décrit comme étant un incident causant le changement d'état ou l'activation d'une action précise. La vitesse inégalée, l'adaptabilité et la fidélité de Qualnet permettent d'optimiser facilement des réseaux existants grâce à la mise en œuvre rapide de modèles et à des outils d'analyse précis.

L'efficacité du noyau Qualnet est qu'il permet d'intégrer facilement de nouveaux protocoles grâce au modèle en couches du simulateur. Chaque protocole exécute sa machine d'états dépendamment de la couche à laquelle il est implémenté. L'occurrence d'un évènement correspond à une transition dans la machine d'états. L'interface entre les couches est aussi basée sur l'occurrence d'un évènement. Chaque protocole peut soit créer des évènements pour transiter à un autre état, ou créer des évènements qui seront traités par un autre protocole. La Figure 4.1 représente la machine d'états d'un protocole dans Qualnet.

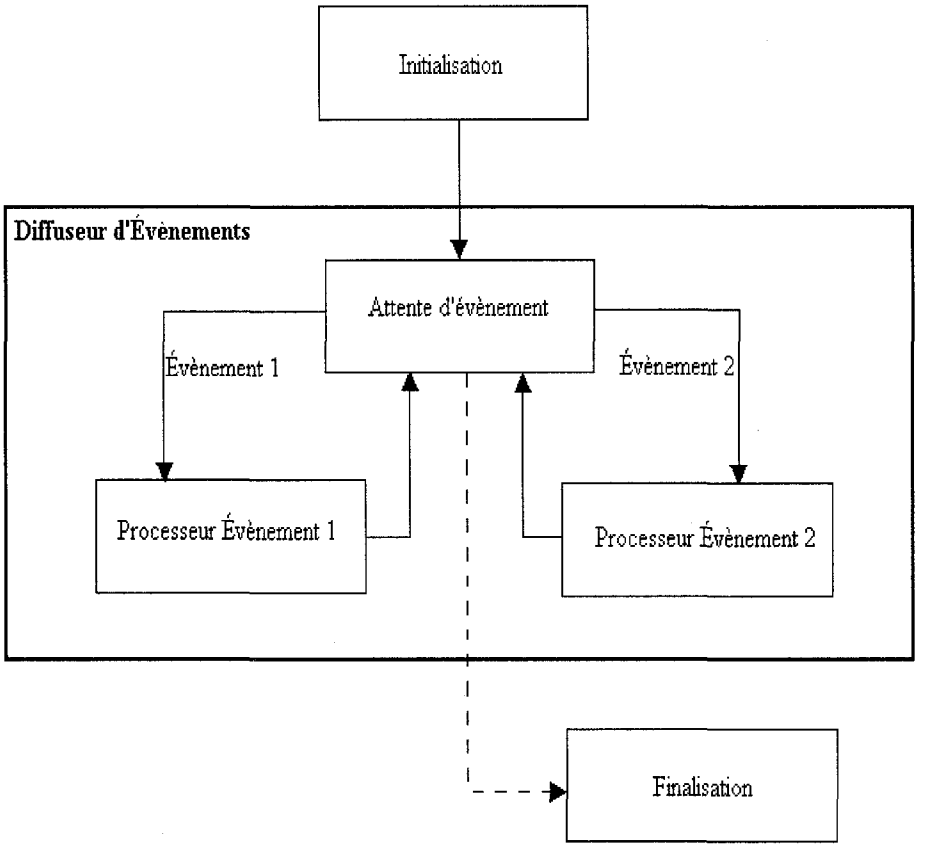


Figure 4.1 Modélisation d'un protocole dans Qualnet

Ainsi, chaque couche est implémentée comme un gestionnaire d'évènement qui reçoit des évènements sous forme de structure de données appelées messages contenant le type d'évènement ainsi que les informations associées. Le gestionnaire d'évènements détermine alors le type d'évènement puis en crée ou non un nouveau. À chaque couche, un protocole est initialisé par une fonction qui reçoit en entrée les données relatives à son fonctionnement et effectue sa configuration. Ainsi, quand un évènement survient, le gestionnaire d'évènements en détermine le type et effectue l'aiguillage vers le protocole concerné. À la fin de chaque simulation, une fonction de finalisation est appelée pour chaque protocole et pour chaque nœud pour l'impression des statistiques. Un évènement

de fin de simulation est généré automatiquement pour permettre la transition vers l'état final des protocoles impliqués dans la simulation. En résumé, chaque protocole doit implémenter les trois fonctions suivantes : une fonction d'initialisation, un gestionnaire d'évènements, et une fonction de finalisation.

4.2 Méthodologie d'implémentation

Cette section présente les détails de l'implémentation du modèle proposé au sein du simulateur. Nous commencerons par justifier le choix de la couche d'implantation du modèle, puis nous arborerons le graphe d'états des différentes composantes de la proposition.

4.2.1 Choix de la couche d'implantation

Dans les réseaux filaires, la découverte de services est résolue au niveau de l'application. Tel qu'indiqué au chapitre II, il a été montré par [7] que dans les réseaux ad hoc l'intégration des deux couches Application et Réseau permet d'augmenter l'efficacité du système. En effet, les auteurs exposent qu'il y aurait plusieurs bénéfices à intégrer la découverte de services au routage. En résumé, l'idée était de gérer de manière la plus efficace possible l'énergie et la mémoire disponible au niveau des terminaux en intégrant la découverte de services au routage. Nous proposons l'intégration du protocole de découverte de service au sein de la couche Application du modèle OSI. Nous croyons que ce choix dans notre proposition offre les mêmes bénéfices que l'alternative d'incorporer la découverte de services au routage. Étant donné que notre proposition est légère en termes de messages de signalisation, la réutilisation de l'infrastructure de routage pour la découverte de services pour diminuer les entêtes de routage aura un impact négligeable sur la charge du réseau pour le protocole que nous proposons.

4.2.2 Graphe d'états

Grâce à la machine d'états, un type d'application comme la découverte de service est naturellement implantable avec Qualnet. Il s'agit de traduire les algorithmes de la Figure 3.7 et 3.9 par un graphe fini d'états, constitué de nœuds correspondant aux évènements et lié par des arcs représentant les transitions. Un graphe d'états doit être borné avec tous ces éléments connectés, et doit respecter des contraintes d'accessibilité et de continuité. La première phase de la proposition, soit la formation du RDV est représenté par le graphe d'états de la Figure 4.2 qui suit.

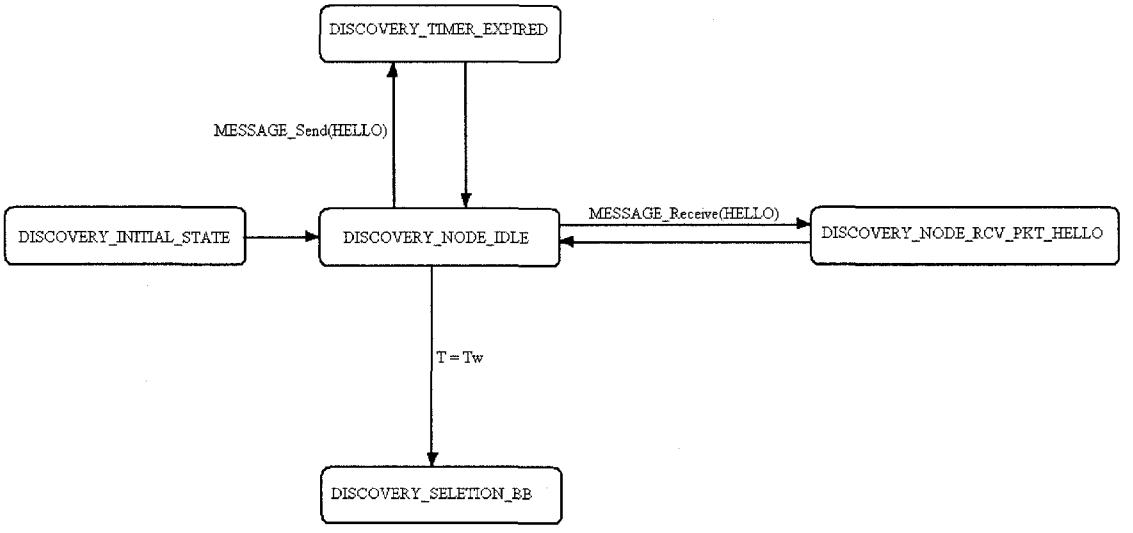


Figure 4.2 Graphe d'états de la phase I

À l'initialisation du protocole, chaque nœud du réseau se trouve dans l'état DISCOVERY_INITIAL_STATE. La fonction DiscoveryInit(Node* node, NodeAddress clientAddr) modélise cet état. Cette fonction s'assure que la structure de données du protocole est formulée au niveau de chaque entité et que les paramètres globaux du protocole sont initialisé. La transition se fait naturellement vers l'état DISCOVERY_NODE_IDLE, où commence la collecte d'informations sur le voisinnage afin que chaque unité puisse collecter les différentes variables requises pour calculer son de stabilité NLFF. La transition vers l'état DISCOVERY_TIMER_EXPIRED représente

l'envoie de messages HELLO entre unités. Cet évènement est modélisé par une structure de données du type MESSAGE_Send(). D'un autre côté, la transition vers l'état DISCOVERY_NODE_RCV_PKT_HELLO, représente la réception d'un message HELLO. Le temps de séjour dans l'état DISCOVERY_NODE_IDLE est de T_w. C'est le temps requis pour que chaque nœud remplit sa table NIT, qui est un membre de la structure de données principale du protocole, soit *dataPtr->NIT*. La transition étiquetée par $T = T_w$ modélise l'évènement chronométré d'atteinte de la fin de ce temps de séjour. Le passage vers l'état DISCOVERY_SELETION_BB représente la fin de la Phase I du protocole.

La suite du graphe d'états s'articule sur le choix qu'aura fait chaque nœud quant à son état. Soit que le nœud est un nœud stable et appartient au RDV, il transite, via l'arc BB=1 vers l'état DISCOVERY_NODE_SERVER_IDLE ou le nœud n'est pas considéré stable et transite via l'arc BB=0 vers l'état DISCOVERY_NODE_CLIENT_IDLE. La Figure 4.3 illustre la suite du graphe d'état du protocole de découverte de services proposé. Les phases II et III du protocole sont modélisés.

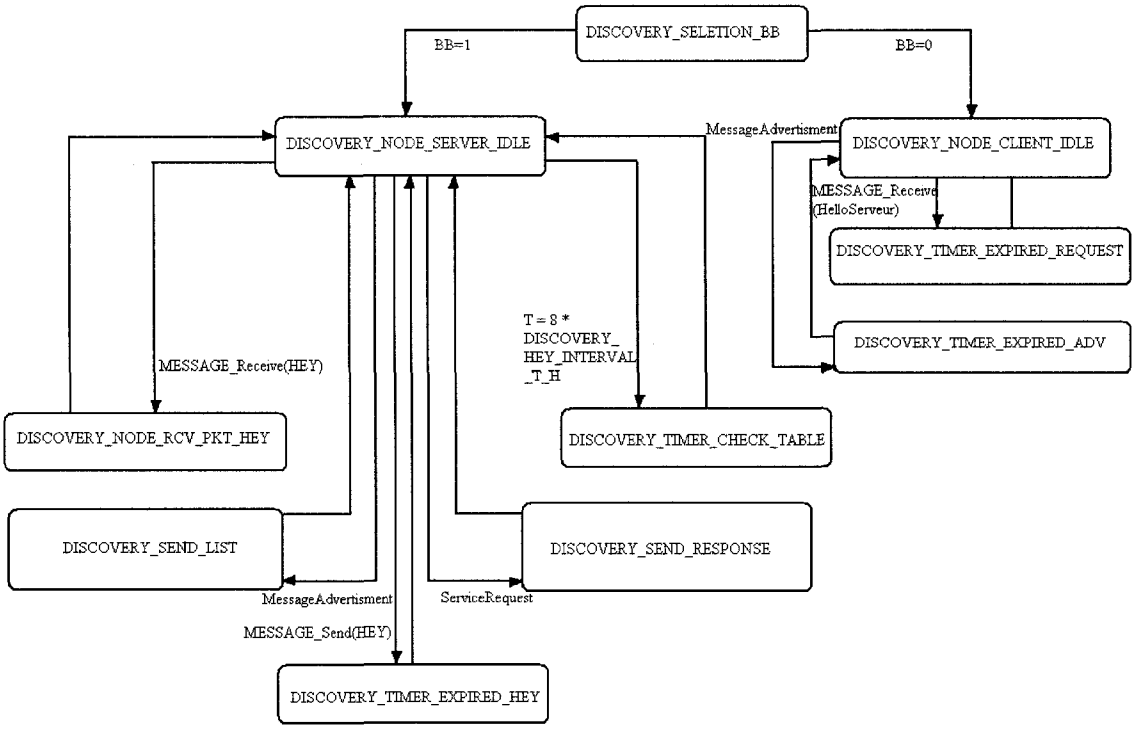


Figure 4.3 Graphe d'états des Phases II et III

État SERVEUR

Le serveur assure la gestion des requêtes des clients. Il se charge principalement de traiter les requêtes des clients et de gérer la liste des services actifs (créer, supprimer, ajouter ou retirer un service), en plus d'changer des messages HEY. Le fonctionnement du serveur se résume à répéter continuellement les deux étapes suivantes :

- attendre une demande de service émanant d'un client;
- traiter la demande de service;
- maintenir le contact avec les nœuds du RDV.

Ainsi, à partir de l'état DISCOVERY_NODE_SERVER_IDLE de la Figure 4.3, l'échange de paquets HEY avec les autres nœuds du RDV est modélisé par l'évènement sur la tansition allant vers l'état DISCOVERY_TIMER_EXPIRED_HEY, soit Envoie HEY. Cet envoie est possible avec l'utilisation de la structure de données MESSAGE_Send(). De la même manière que pour la réception de messages HELLO, la

transition allant vers l'état DISCOVERY_NODE_RCV_PKT_HEY est étiquetée par l'évènement Reçoit HEY. À la réception de cet évènement, pour des mesures de maintenance, chaque nœud du RDV doit enregistrer le temps de réception du paquet HEY, soit *dataPtr->lastHeard*. Cette étape correspond à la Phase II de la proposition, soit la maintenance du RDV. La transition allant vers l'état DISCOVERY_TIMER_CHECK_TABLE modélise le chronomètre ayant atteint le temps T = 8 * DISCOVERY_HEY_INTERVAL_T_H.

La dernière Phase du protocole est prise en charge par les clients et les serveurs. Du côté des serveurs, à partir de l'état DISCOVERY_NODE_SERVER_IDLE de la Figure 4.3, à la réception d'un évènement MessageAdvertisment qui correspond à la publication de service formulée par un client, le serveur transite vers l'état DISCOVERY_SEND_LIST. À ce niveau, le serveur effectue une suite de vérification avant d'envoyer le nouveau service enregistré à sa liste, *listeServeurs*, contituée des nœuds du RDV. D'un autre côté, la transition étiquetée par l'évènement ServiceRequest correspond à la requête envoyée par un client. Le serveur transite alors vers l'état DISCOVERY_SEND_RESPONSE.

État CLIENT

À partir de l'état DISCOVERY_NODE_CLIENT_IDLE de la Figure 4.3, si le client offre un service, il formule un évènement MessageAdvertisment et passe dans l'état DISCOVERY_TIMER_EXPIRED_ADV. Il y séjourne juste pour le temps que ça prend pour qu'un nœud du RDV réponde à son offre. D'un autre côté, après réception d'un HelloServeur et de retour dans l'état précédent, le client peut vouloir émettre une requête de service et ainsi se mettre en attente de la réponse à sa demande en passant dans l'état DISCOVERY_TIMER_EXPIRED_REQUEST. Cette séquence donne un apreçu et complète les trois phases du protocole de découverte proposé.

4.3 Plan d'expériences

La définition du plan d'expériences est une tâche qui s'est avéré ardue et qui demande des préalables consistants. Dans cette section, nous présenterons d'abord la configuration de la simulation pour ensuite faire un survol rapide des tests préliminaires et des observations que nous en avons tirées. Nous continuerons sur une présentation des indices de performance et nous argumenterons le choix des facteurs primaires que nous avons identifiés. Enfin, seront présentés les sessions retenues pour notre plan d'expériences.

4.3.1 Configuration de la simulation

Il faut évoquer ici le manque de standards dans les simulations des réseaux ad hoc. La définition de ces standards est un champ plus ou moins actif de la recherche mais il n'existe à l'heure actuelle aucun consensus sur la manière dont il faudrait conduire des simulations MANET. Nous nous sommes surtout attachés à faire des expériences qui nous permettent de comparer notre modèle avec ceux proposés dans la littérature.

Dans le cadre de notre série de simulations, nous considérons un réseau mobile ad hoc uniformément distribué sur une surface carré ou rectangulaire. Les unités mobiles utilisent une interface de communication sans fil de type 802.11b en mode ad hoc avec un débit d'interface de 2 Mbps. La propagation des signaux se fait suivant un modèle en espace libre. Le modèle de mobilité choisi pour les simulations est le *Random-Waypoint* déjà largement utilisé dans la littérature, car approchant plus la réalité. En effet, dans ce modèle, une unité mobile choisit une destination à l'intérieur de la surface définie pour ensuite se déplacer en direction de sa destination à vitesse constante choisie selon une distribution uniforme située entre MIN_SPEED et MAX_SPEED. Une fois que l'unité atteint sa destination, elle attend sur place pour un temps déterminé par PAUSE_TIME avant de choisir une nouvelle destination et de s'y rendre. Deux variables permettent de contrôler le mouvement des unités mobiles : la vitesse v et la durée de pause p. Le temps

de simulation sera de 10 minutes. Le nombre de nœuds dans le réseau sera varié et le temps entre les requêtes de services sera fixé à 10s. Le protocole de routage utilisé sera DSR.

4.3.2 Tests préliminaires

La génération de l'univers logiciel de notre simulation nous a paru complexe car les paramètres à régler étaient nombreux. Des tests préliminaires nous ont toutefois permis de constater l'influence négligeable sur nos indices de performance de quelques paramètres.

Le premier des paramètres correspond à l'intervalle de temps T_H. La phase I commence avec la première étape qui est la découverte du voisinage. Pendant T_w secondes, les nœuds s'envoient des messages HELLO à chaque T_H secondes pour découvrir les autres nœuds de leur entourage et former ainsi leur NIT. Plus la fréquence d'envoie de messages Hello est élevée, T_H petit, plus il y aura de paquets qui seront envoyés dans cet intervalle de temps Tw. Le but de l'envoie des messages Hello est de former la NIT. La fréquence d'envoie de messages Hello n'avait aucun effet sur les indices de performance retenus.

L'intervalle de temps T_W correspond à la période pendant laquelle les nœuds s'envoient des messages HELLO à chaque T_H secondes pour découvrir les autres nœuds de leur entourage et former ainsi leur NIT. Plus la fréquence d'envoie de messages Hello est élevée, T_H petit, plus il y aura de paquets qui seront envoyés dans cet intervalle de temps Tw. De la même manière, l'intervalle de temps T_W n'affectait pas les indices de performance à l'étude.

Le protocole de routage utilisé par les nœuds a aussi fait l'objet d'une analyse préliminaire. Les clients envoient des requêtes de service par procédure de «broadcast» avec une durée de vie fixe. Le «broadcast» se fait dans la surface de propagation du nœud. L'architecture proposée déploie des nœuds un peu partout dans le réseau. Quelques nœuds forment le réseau dorsal virtuel et ont comme tâche principale de

répondre aux requêtes de services envoyées par les clients. On prévoie que lorsqu'un client envoie sa requête il réussira à rejoindre au moins un nœud du BB. Ce dernier lui enverra un serviceReply par unicast. Les résultats ont montrés que peut importe le protocole de routage utilisé, les impacts sur les indices de performances étaient négligeables. Cet état de fait, certes intéressant, nous a permis de nous concentrer sur des paramètres généraux plus significatifs.

4.3.3 Indices de performance

Il s'agit à présent de déterminer les indices de performance qui permettent d'évaluer l'architecture de découverte de service proposée. Plusieurs paramètres déterminent la manière dont un système opère. Le but est de déterminer pour un environnement particulier l'ensemble de paramètres le plus convenable. C'est ainsi qu'on valide l'exactitude du protocole en s'assurant qu'il fonctionne comme voulu. Cela nous permet d'évaluer la performance et le comportement de notre protocole. L'accent est mis sur les critères et les résultats. Une stratégie de découverte de services est dite performante si elle possède ces 3 caractéristiques :

- le premier indice est la charge réseau en termes de nombre de paquets. Cet indice correspond au poids du protocole sur le réseau;
- le second indice est le taux moyen de succès. Ce taux est le ratio du nombre de tentatives réussites sur le nombre total de requêtes envoyées. Une tentative est considérée réussite quand le client reçoit dès le premier envoie une réponse à sa requête. Ainsi, dans notre algorithme, le client n'envoie pas une autre requête jusqu'à ce qu'il reçoive une réponse;
- le dernier indice est le délai moyen entre le temps qu'un client envoie une requête et le temps que ce dernier reçoive une réponse. Cet indice est particulièrement important pour des applications en temps réel. Cette métrique est particulièrement importante quand des applications en temps réel attendent pour un service.

Ces indices de performances dépendent de plusieurs facteurs. Le nombre de paramètre pouvant à priori influencer un indice de performance peut être très grand. Ainsi, le coût d'une expérience augmente avec le nombre de facteurs à considérer, d'où la nécessité d'une sélection soigneuse des facteurs. Nous présentons dans ce qui suit les facteurs primaires retenus.

4.3.4 Facteurs primaires

Le Tableau 4.1 de la page suivante présente les facteurs primaires retenus et leurs niveaux associés. La mobilité des nœuds est de toute évidence un facteur dont l'influence est intéressante à analyser. Nous avons retenus trois ordres de grandeur pour des vitesses allant de 1m/s à 20m/s. Un autre facteur est le nombre de serveurs dans le réseau. Ce nombre correspond en fait au nombre de nœuds qui offrent un service et qui doivent envoyer des publications de service pour ce faire connaître. Le nombre de serveurs est un facteur d'importance et les différents niveaux considérés sont de 1, 3 et 5 serveurs. Le nombre de clients correspond au nombre de nœuds dans le réseau demandant un service. Un client formule une requête de service et attend pour la réponse. Nous avons choisi trois niveaux avec comme niveau le plus élevé 30 clients par rapport au plus bas niveau qui est de 10 clients. La topologie du terrain est un autre facteur d'envergure dans les réseaux ad hoc. C'est pourquoi nous avons choisi des dimensions de terrain allant de la forme rectangulaire (1500m*300m) à la forme carrée (2400m*2400m). Le dernier facteur à l'étude sera le nombre de nœuds appartenant au BB. Dans l'optique de vérifier expérimentalement les bornes de l'équation (3.8) donnée dans le chapitre III, l'étude de ce facteur paraît indispensable. Le nombre de nœuds appartenant au BB variera entre (8 et 55) nœuds.

Tableau 4.1 Facteurs primaires et niveaux associés :

Facteurs		Niveaux	
Nom	**Symbole**	**Nom**	**Description**
Nombre de nœuds dans le réseau	N	Petit Moyen Grand	50 80 150
Mobilité des nœuds	V	Petit Moyen Grand	(1,4)m/s (8,10)m/s (15,20)m/s
Nombre de serveurs	S	Petit Moyen Grand	1 3 5
Nombre de clients	C	Petit Moyen Grand	10 20 30
Topologie du terrain	T	Petit Moyen Grand Très grand	(1500m*300m) (1500m*1500m) (2400m*1500m) (2400m*2400m)
Nombre de nœuds appartenant au RDV	R	Petit Moyen Grand	(8,12,15)nœuds (20,25,28)nœuds (35,40,55)nœuds

4.3.5 Sessions

Pour déterminer l'influence des facteurs primaires sur la performance de la proposition, des expériences à sessions multiples sont nécessaires. Nous avons fait le choix d'effectuer des expériences «un facteur à la fois». Les sessions ou scénarios à l'étude se résument comme suit :

- pour étudier l'effet de la mobilité des nœuds, on fera varier la valeur de p, le temps de pause, dans le modèle de mobilité *Randon_Waypoint*, (0, 50, 100, 250, 500 et 900) secondes. La valeur de v, vitesse, sera maintenue à 20m/s. Il est à remarquer que plus la valeur de p est petite plus la mobilité sera élevée. Tous les autres paramètres resteront constants : Tsim = 900S, Tw= 1S, TH = 0.1S, topologie rectangulaire (1500m * 300m), 3 serveurs, le nombre de nœuds dans le réseau (50, 80, 150) nœuds, le protocole de routage DSR;
- pour étudier l'effet du nombre de serveurs sur le délai moyen de réponse à une requête, le nombre de serveurs variera entre (1, 3, 5, 10) serveurs, quand tous les autres paramètres resteront constants : Tsim = 900S, Tw= 1S, TH = 0.1S, v = 20m/s, p = 200s, (10, 15, 20) clients, THEY=10s, topologie rectangulaire (1500m * 300m). Pour chaque nombre de serveurs, quatre expériences sont effectuées avec le nombre de nœuds dans le réseau (30, 40, 50, 70), le protocole de routage DSR. Dans chaque expérience, pour chaque session, le nombre de clients est sélectionné parmi, 10, 15 et 20. Ainsi, chaque point sur le graphique correspond à une moyenne de 12 scénarios;
- pour étudier l'effet du nombre de clients, ce dernier variera entre (1 et 60), quand tous les autres paramètres resteront constants : Tsim = 900S, Tw= 1S, TH = 0.1S, v = 20m/s, p = 200s, THEY = 10s, topologie rectangulaire (1500m * 300m). Pour chaque nombre de clients, cinq expériences sont effectuées avec le nombre de nœuds dans le réseau (40, 50, 80, 90), le protocole de routage DSR. Dans chaque expérience, pour

chaque session, le nombre de serveurs est sélectionné parmi, (3, 4, 5, 6, 7, 8). Ainsi, chaque point sur le graphique correspond à une moyenne de 12 scénarios;

- pour étudier l'effet de la topologie, on déploiera 100 nœuds dans une zone rectangulaire moyenne 1500m*300m et grande 2400m*1500m et dans une zone carrée moyenne 1500*1500m et grande 2400m*2400m. Pour des vitesses de (1,4,8,15,20)m/s et p = 0, tous les autres paramètres resteront constants : Tsim = 900S, Tw= 1S, TH = 0.1S, THEY = 10s, 3 serveurs, 20 clients, le protocole de routage DSR;
- pour étudier l'effet de la densité du réseau dorsal virtuel sur le délai moyen de réponse à une requête, le nombre de nœuds appartenant au BB variera entre (8 et 55), quand tous les autres paramètres resteront constants : Tsim = 900S, Tw= 1S, TH = 0.1S, v=10m/s, p = 0s, THEY = 10s, topologie rectangulaire (1500m * 300m), 3 serveurs, 10 clients. Pour chaque densité de nœuds, cinq expériences sont effectuées avec le nombre de nœuds dans le réseau (40, 60, 80,100), le protocole de routage DSR.

Tel que précisé, un nombre d'observations doivent être collectées pour chaque quantité considérée, de telle sorte que les distributions de ces quantités et leurs moments peuvent être estimés avec une précision suffisante. Ainsi, pour approcher des intervalles de confiance de 95% dans les mesures, nous effectuerons une série de simulations pour chaque indice de performance.

4.4 Analyse des résultats

Une fois les indices de performance et les sessions définis, on peut mener les expériences et procéder à leur interprétation. Nous nous intéressons à l'évolution du système sur les indices de performance identifiés. Dans nos représentations graphiques,

c'est une moyenne qui est représentée en ordonnée de nos graphes qui présentent les résultats.

Dans les sous-sections suivantes, nous discuterons de l'effet sur l'évolution du système des facteurs primaires considérés. L'influence de la mobilité, du nombre de serveurs, du nombre de clients, de la topologie du réseau et du nombre de nœuds appartenant au RDV seront ainsi successivement étudiés dans leur impact sur le délai moyen de réponse à une requête, sur le taux moyen de succès et sur la charge du réseau.

4.4.1 Influence des facteurs sur le délai moyen

Effet de la mobilité sur le délai moyen de réponse

Le premier facteur à l'étude est la mobilité des nœuds du réseau. On note, à partir des résultats illustrés à la Figure 4.4, que lorsque la mobilité des nœuds est très élevée, p<300s, le délai moyen de réponse à une requête est plus élevé. Ce temps se stabilise pour des vitesses v moyenne et faible. Les 3 scénarios sont ordonnés suivant le nombre de nœuds dans le système. On constate que pour des densités de nœuds dans le réseau, petite, moyenne et élevée (50, 80 et 100 nœuds), le délai de réponse a augmenté de façon non-significative. Ces courbes modélisent bien le comportement de notre approche face à la mobilité des nœuds. Nous trouvons très acceptable les délais reportés car ils relatent d'une stratégie où les délais de réponse sont peut affectés par la mobilité des nœuds. Le plateau dans la figure est très significatif dans le sens que peut importe où les unités mobiles se trouvent dans la topologie, ils réussissent à trouver un nœud du BB qui répondra la leur requête. Ce délai de réponse est le même pour une vitesse moyenne ou élevée. Même pour une densité de réseau élevé, nous avons là une même allure de courbe et nous estimons que les résultats sont très acceptables.

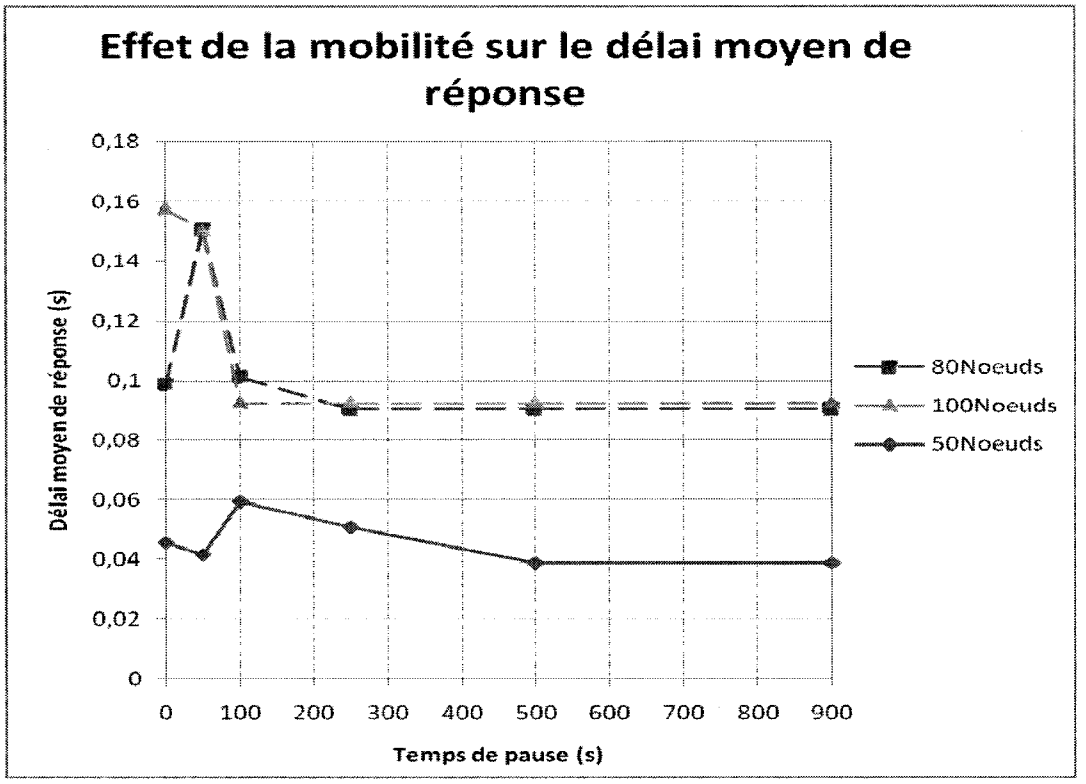


Figure 4.4 Effet de la mobilité sur le délai moyen de réponse

Pour comparer notre proposition avec les autres architectures, la Figure 4.5 montre les courbes de tendance du délai moyen de réponse en fonction des différentes valeurs de *p*. Notre proposition est comparée au protocole DSDP [19] et ce, pour un scénario constitué de 3 serveurs avec le nombre de clients 10 et 30 et une topologie rectangulaire de (1500m*300m).

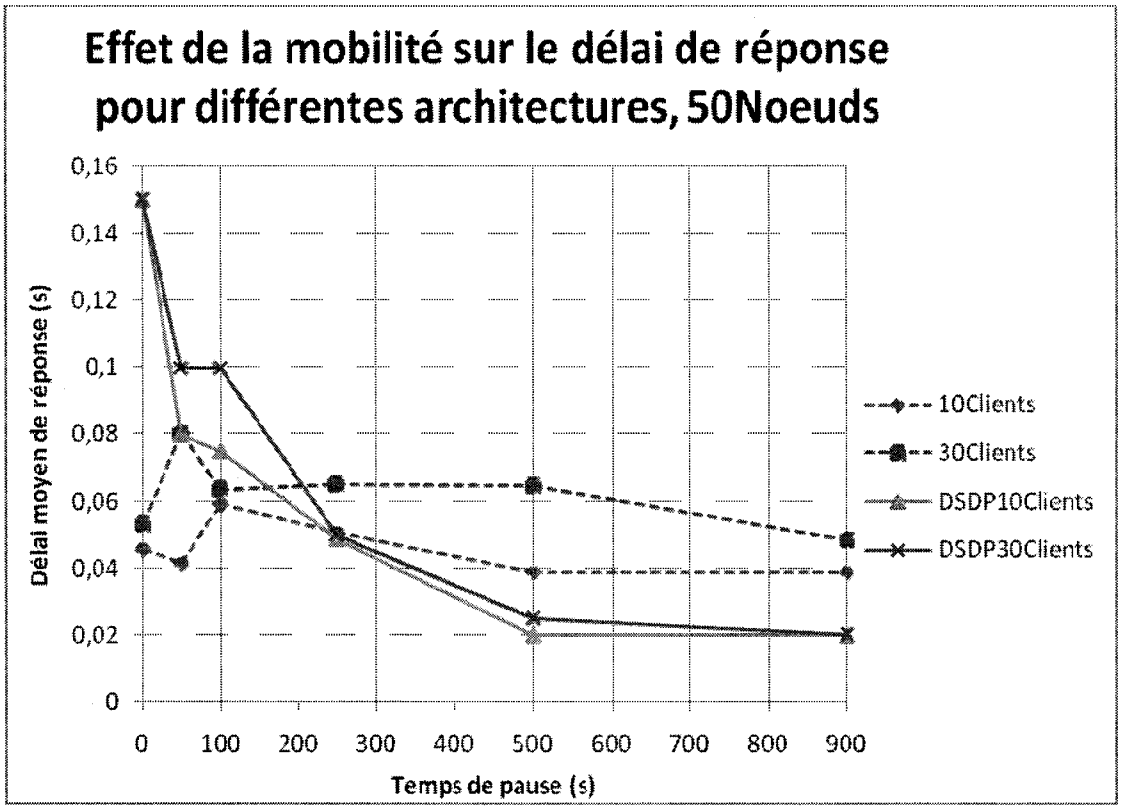


Figure 4.5 Comparaison de l'effet de la mobilité sur le délai moyen de réponse avec DSDP pour les cas de 3-serveurs/10-30 clients

Pour des temps de pause $p < 250$, nos courbes se trouvent en dessous de celles de la proposition DSDP. Ceci montre que notre proposition est plus performante en termes de délai de réponse à une requête pour une mobilité de nœuds élevé. Les courbes DSDP suivent une même allure que nos courbes avec un plateau à partir de vitesses moyennes et faibles. Nous considérons insignifiante la différence de vitesses entre nos propositions pour une faible mobilité. Ceci peut être expliqué par l'environnement considéré. En effet, en s'approchant des temps de pause de $p = 900$s, on s'approche vers des réseaux immobiles, stables où la meilleure approche est la centralisation, là où un seul serveur maintienne la liste des services. Le déploiement de nœuds stables dans le réseau qu'on propose devient exigeant dans un tel environnement. Cependant, pour des réseaux mobiles comme des réseaux ad hoc, l'avantage de notre proposition est que le délai de

réponse à une requête pour des nœuds très mobiles est meilleur que ceux proposés dans la littérature.

Effet du nombre de serveurs

Le deuxième facteur à l'étude est le nombre de serveurs, c'est-à-dire le nombre de nœuds offrant des services. Un nœud qui offre un service doit publier ce dernier. Il envoie un message de publication. Quand un nœud du BB reçoit ce message, il y répond par un message HelloServeur. Si le serveur n'a pas reçu ce message après un certain temps, *Tattenteserveur*, il envoie une autre requête de service. Et ainsi de suite jusqu'à ce que son service soit publié. Le nœud du BB qui a reçu la publication doit insérer le service offert dans sa table *servicecache* et avertir les autres nœuds du réseau dorsal virtuel de l'enregistrement qu'il vient de recevoir en envoyant des messages de publication publiée. À première vue, on avait supposé que le nombre de serveurs aura un impact négatif sur le délai de réponse aux requêtes. Contrairement à notre hypothèse, le nombre de serveurs dans le réseau n'affecte pas le temps de réponse à une requête. Dans la Figure 4.6, les courbes sont très peu affectées par l'augmentation du nombre de serveurs. Ceci peut être compris par le fonctionnement du protocole. En effet, le processus d'enregistrement et de requête de service se fait via le réseau dorsal virtuel, BB. C'est-à-dire à travers des nœuds stables éparpillés un peu partout dans l'architecture. Une fois qu'un nœud du BB intercepte une publication, il envoie l'enregistrement aux autres nœuds du BB. Cet envoie se fait en unicast parce que les nœuds du BB se connaissent. Cette procédure ne surcharge pas le réseau, ne causant ainsi pas de congestion, ce qui facilite le fait que le réseau peu supporter plusieurs serveurs. Sachant que c'est la surcharge du réseau qui augmente le délai de réponse à une requête, on conclue que le nombre de serveurs n'affecte pas le délai de réponse à une requête.

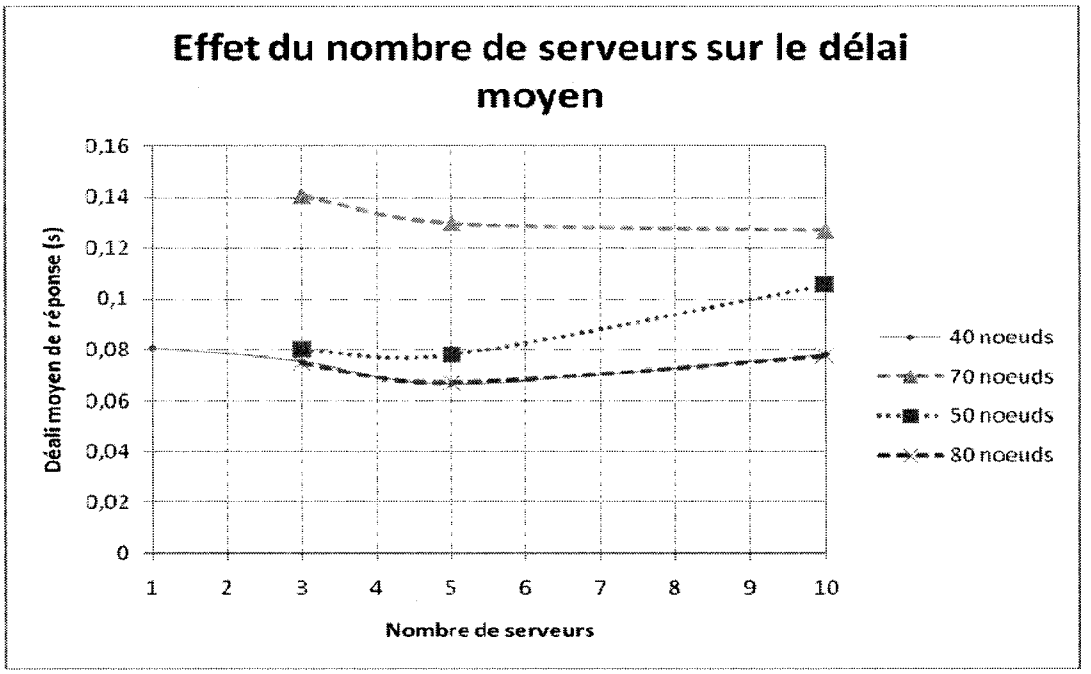


Figure 4.6 Effet du nombre de serveurs sur le délai de réponse

Dans la Figure 4.7 notre proposition est comparée au protocole DSDP et les courbes de tendance du délai moyen de réponse en fonction de la vitesse des nœuds y sont illustrées.

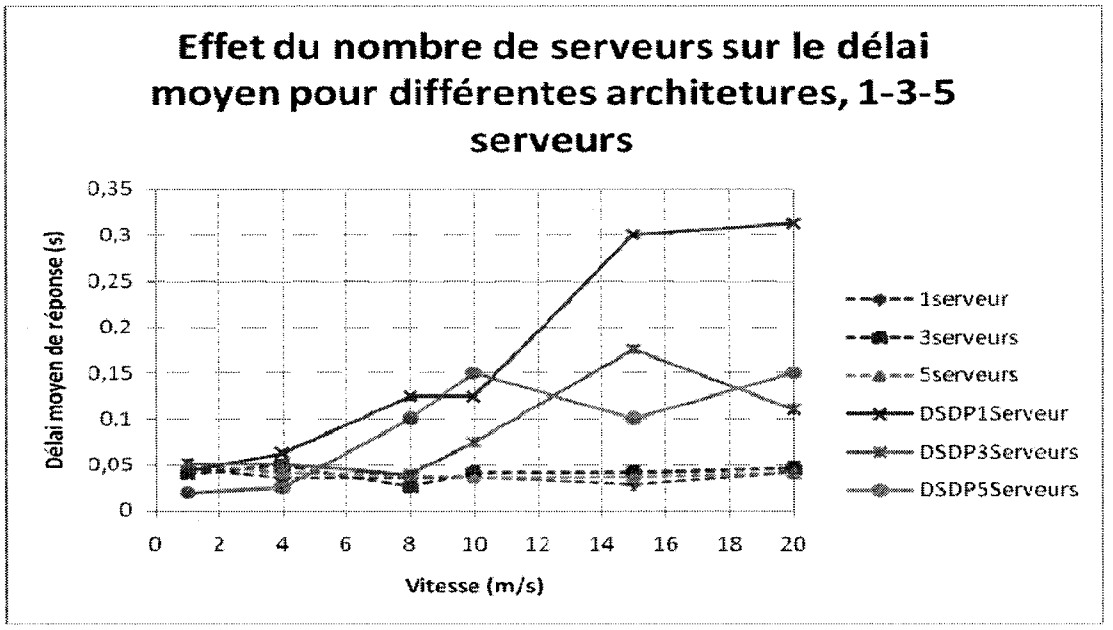


Figure 4.7 Comparaison de l'effet du nombre de serveurs sur le délai moyen de réponse pour les cas de 1,3,5 serveurs / 10 clients.

Nous constatons que nos trois courbes se trouvent à avoir des délais de réponse plus faibles que la proposition DSDP. On vante le fait que notre proposition répond mieux que les autres architectures proposées dans le cas où le réseau offrait plusieurs services. Avec l'augmentation du nombre de serveurs, le temps de réponse est resté le même par rapport à l'autre proposition où il s'est trouvé à croître avec l'accroissement du nombre de serveurs. Cette situation est due au fait que la disponibilité dans le réseau de nœuds judicieusement choisis permet la collecte efficace de messages de Publication. Dans l'approche DSDP, un nœud stable utilise un mécanisme de multicast pour avertir de la présence d'un service dans le réseau. Dans notre approche, le routage se fait par unicast ce qui est moins coûteux et les courbes le témoignent.

Effet du nombre de clients

Le troisième facteur à l'étude est le nombre de clients désirant des services. Les clients envoient des requêtes de service par procédure de «broadcast». Le «broadcast» se fait dans la surface de propagation du nœud, et ceci dans un diamètre d'à peu près 250m,

en plus d'avoir un nombre de sauts fixe. L'architecture proposée déploie des nœuds un peu partout dans le réseau. Ces nœuds forment le réseau dorsal virtuel et ont comme tâche principale de répondre aux requêtes de services envoyées par les clients. On prévoie que lorsqu'un client envoie sa requête il réussira à rejoindre au moins un nœud du BB. Ce dernier lui enverra un message serviceReply par unicast. On pense que plus le nombre de clients sera élevé, plus le nombre de messages de ServiceRequest dans le réseau sera élevé. En effet, les courbes de la Figure 4.8 confirment cette hypothèse, le délai de réponse à une requête augmente de façon régulière avec le nombre de clients dans le réseau.

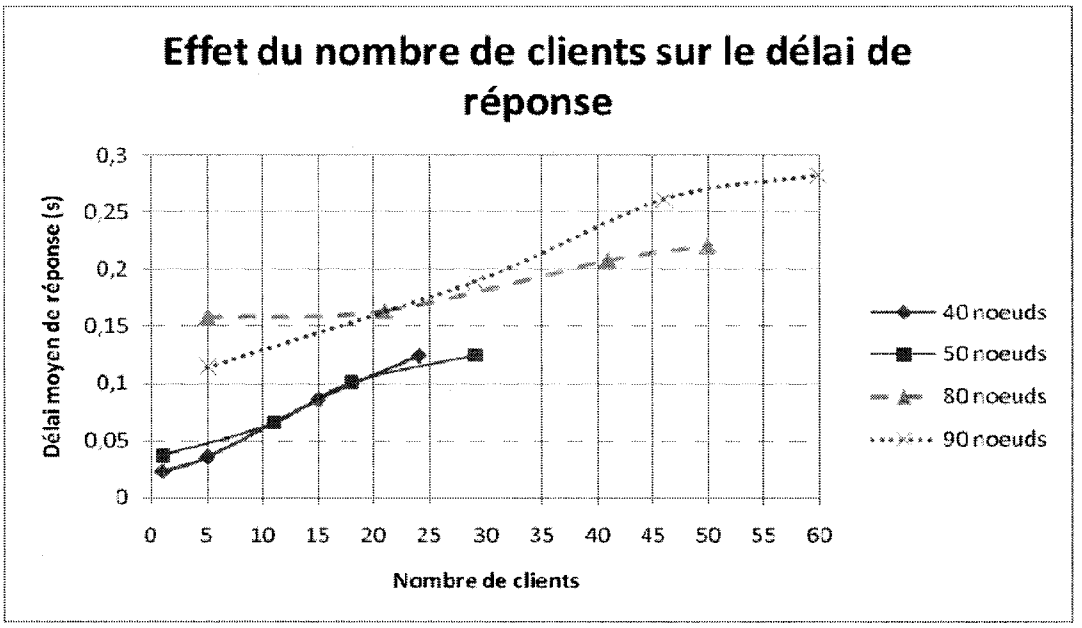


Figure 4.8 Effet du nombre de clients sur le délai de réponse

Pour comparer notre proposition avec l'architecture DSDP, la Figure 4.9 montre les courbes de tendance du délai moyen de réponse en fonction de la vitesse des nœuds.

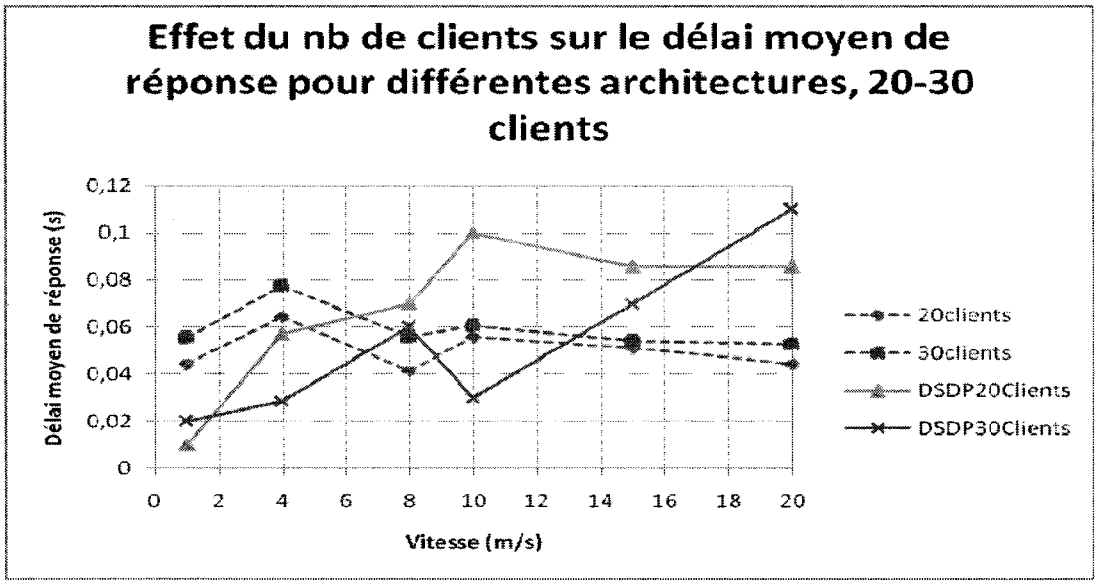


Figure 4.9 Comparaison de l'effet du nombre de clients sur le délai moyen de réponse avec d'autres protocoles pour les cas de 3 serveurs / 20-30 clients

Ainsi, on note des variations irrégulières dans les deux protocoles. On ne peut conclure une tendance pour les courbes. En particulier, on remarque que les délais de réponse de notre proposition se situent entre (0,04 et 0,08) secondes. Pour la proposition DSDP, les délais s'étalent de (0,01 à 0,11) secondes. Ces résultats sont peu attirants pour DSDP car l'approximation des courbes irrégulières (DSDP20Clients et DSDP30Clients) par une équation de degré un, produit des pentes croissantes par rapport à l'approximation de nos courbes ou les pentes sont quasi régulières. On en conclue pour notre proposition que le nombre de clients dans le réseau à peu d'impact sur le délai moyen de réponse à une requête.

Effet de la topologie du réseau

L'autre facteur à l'étude est la topologie du réseau considéré. La Figure 4.10 qui suit montre l'effet de la topologie sur le délai moyen de réponse.

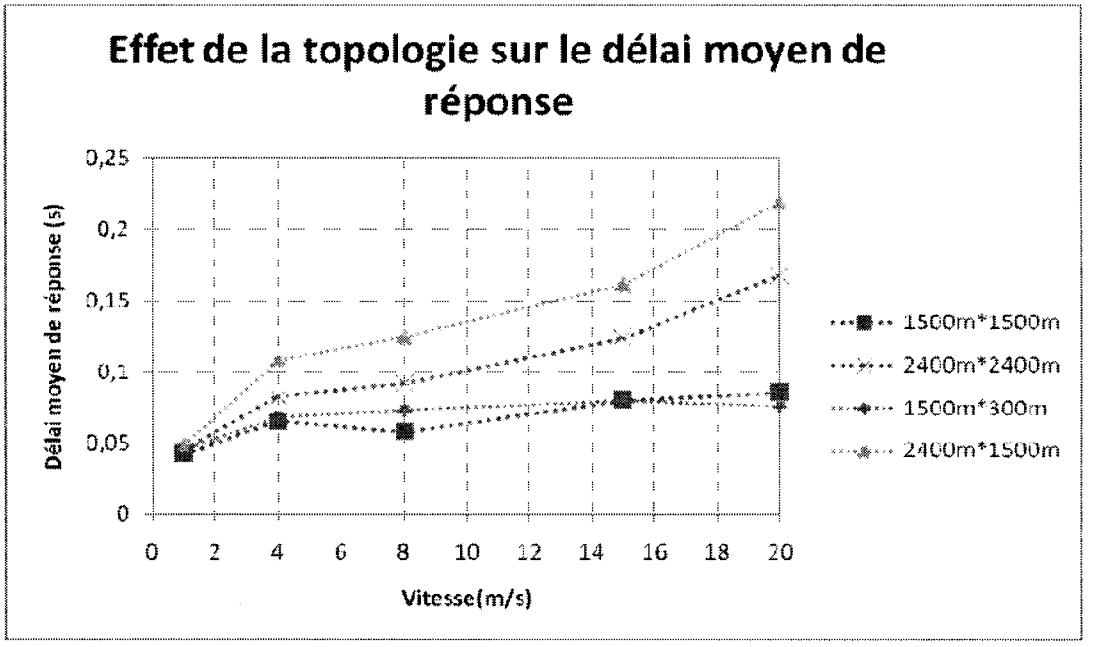


Figure 4.10 Effet de la topologie sur le délai moyen de réponse

On observe que le délai de réponse croît de manière assez lente lorsque la topologie devient plus grande. Cette augmentation est peu significative pour des terrains de surface petite et moyenne. Cependant, pour des grandes et très grandes topologies, cette augmentation devient significative. À notre surprise, la très grande topologie (2400m*2400m) a enregistré de meilleure temps que celle de (2400m*1500m).

Effet de la densité du réseau dorsal virtuel

Le dernier facteur à l'étude est le nombre de nœuds appartenant au BB. Tel qu'indiqué, le BB, réseau dorsal, est formé de nœuds ayant comme principale tâche de répondre aux requêtes de service initiées par les clients. Ils maintiennent une liste de service et agissent ainsi comme des serveurs distribués dans le réseau. La caractéristique qui distingue ces nœuds des autres est leur facteur de stabilité. On suppose que plus le réseau dorsal est dense, constitué de plusieurs nœuds, plus le temps de réponse à une requête de service sera petit. On pense qu'il sera beaucoup plus probable que le client trouve un nœud du BB dans sa surface de transmission et aura par ce conséquent une

réponse immédiate à sa requête. La Figure 4.11 qui suit illustre l'effet de la densité du BB sur le délai moyen de réponse à une requête.

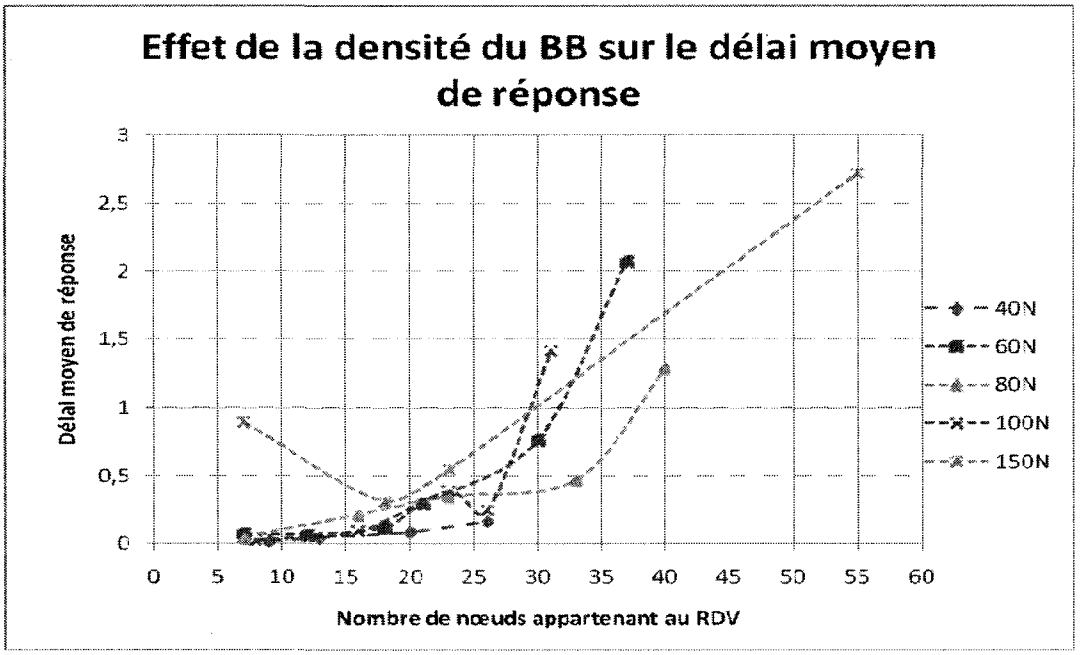


Figure 4.11 Effet de la densité du BB sur le délai moyen de réponse

On remarque que le réseau sature pour une densité de nœuds supérieure à 30. Plus précisément, si le nombre de nœuds appartenant au BB dépasse cette valeur de densité, les temps de réponse aux requêtes de service deviennent intolérables. D'un autre côté, les simulations indiquent que peu importe le nombre de nœuds dans le réseau, un RDV constitué de 15 à 25 nœuds offre des délais de réponse admissibles. Il fallait absolument avoir recours aux expériences pour repérer dans les résultats de simulation, le point de convergence recherché. Cette conclusion n'a pu être formulée sans les simulations. Ainsi, on peut finalement confirmer l'exactitude des bornes inférieures et supérieures de l'équation (3.8) formulée au chapitre III.

4.4.2 Influence des facteurs sur le taux moyen de succès

Effet de la mobilité sur taux

La Figure 4.12 montre l'effet de la mobilité sur le taux moyen de succès des tentatives. On rappelle que cet indice correspond au ratio du nombre de tentatives réussites sur le nombre total de requêtes envoyées. Cette mesure est faite dans le but d'évaluer l'efficacité de l'approche.

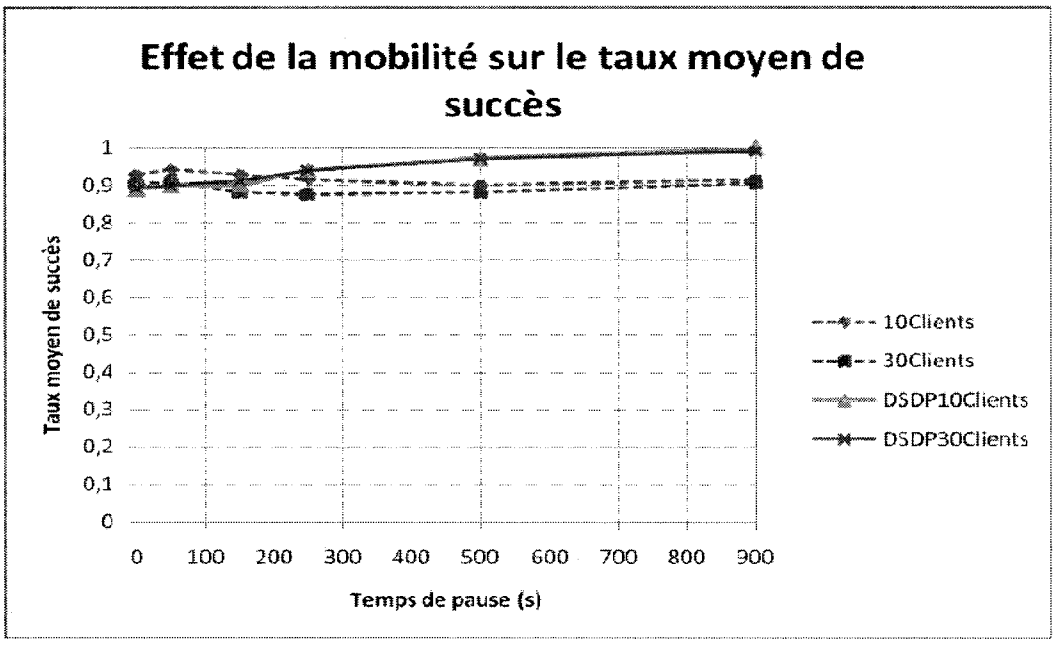


Figure 4.12 Effet de la mobilité sur le taux moyen de succès des requêtes

Le taux moyen de succès ne connaît pas de variation significative par rapport à la mobilité des nœuds. Le taux de succès aux requêtes envoyés pour les deux approches est très similaire. Cette situation s'explique par le fait que les deux approches sont basées sur le déploiement d'un ensemble de nœuds stables dans le réseau pour répondre aux requêtes. Dans les deux modèles se trouvent des unités dispersées. Dans le cas où il existe une seule entité qui soit sollicité en abondance, le taux de succès sera affecté de façon significative. Dans notre proposition, les taux de moyen de réponse aux requêtes ne connaissent aucune variation notable par rapport à la vitesse des nœuds.

Effet du nombre de serveurs et clients

La Figure 4.13 montre l'effet du nombre de serveurs sur le taux moyen de succès. Encore une fois, on s'aperçoit que dans l'approche DSDP, comme pour le délai moyen de réponse à une requête, indice de performance étudié dans la sous-section précédente, le taux moyen de réponse à une requête est très affecté par le nombre de serveurs dans le réseau. Comparativement, notre proposition offre des pourcentages de succès très similaires et peu variants avec la mobilité peu importe le nombre de serveurs.

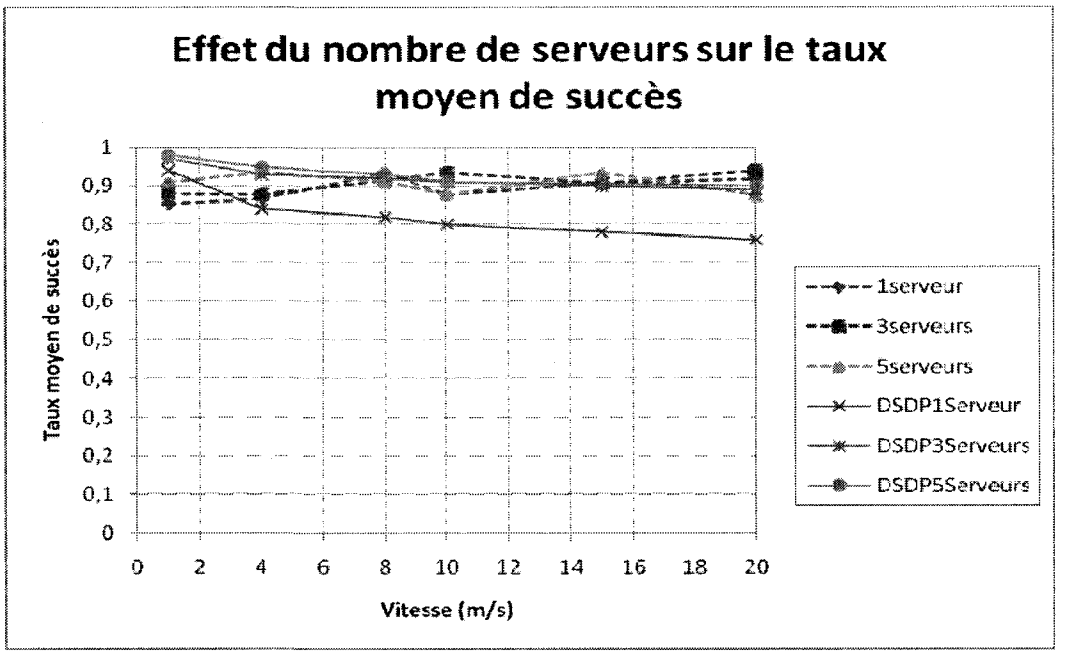


Figure 4.13 Effet du nombre de serveurs sur taux

La Figure 4.14, quant à elle, illustre l'effet du nombre de clients sur le taux moyen de succès. De la même manière, notre proposition offre des taux de succès stables et variant peu avec la mobilité des nœuds.

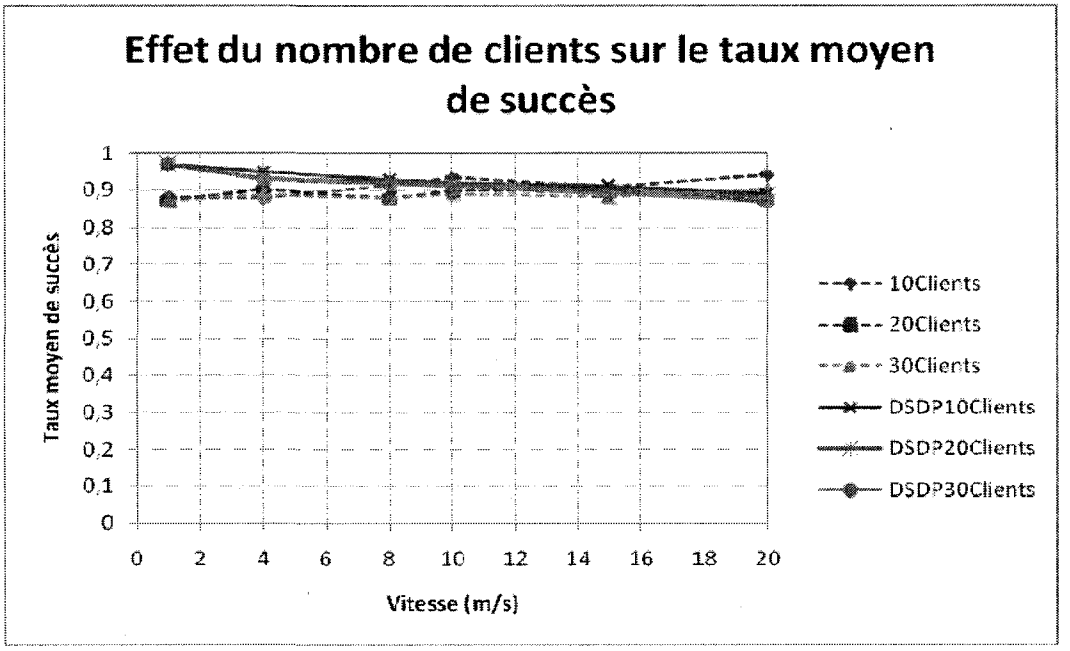


Figure 4.14 Effet du nombre de clients sur taux

Effet de la topologie

La Figure 4.15 qui suit présente les courbes comparatives de l'effet de la topologie du terrain sur le taux moyen de succès des requêtes.

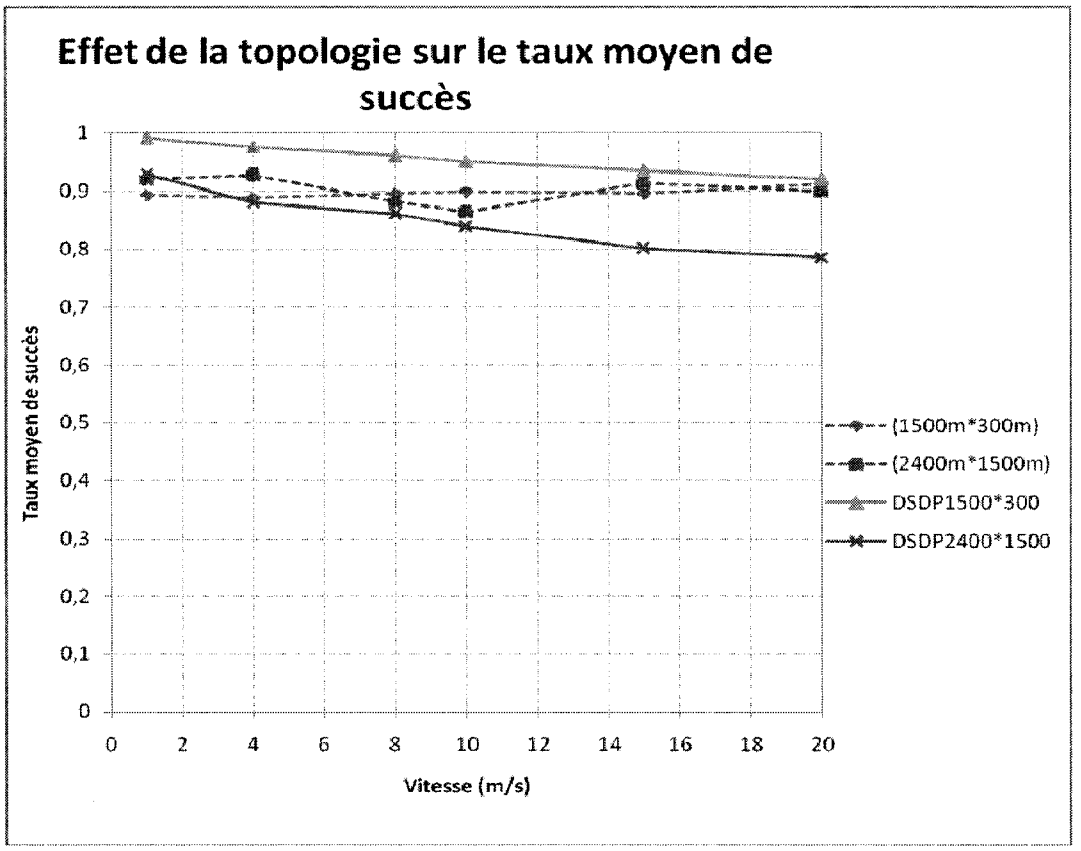


Figure 4.15 Effet de la topologie sur le taux moyen de succès

Ces résultats montrent, une fois de plus, comme pour les autres facteurs étudiés, cet indice de performance qu'est le taux moyen de succès est peut ou pas affecté par la topologie. D'un autre côté, la performance de l'approche DSDP, est dégradé pour des topologies de grande surface, soit (2400m*1500m). On note des résultats aussi bas que 80%.

Effet de la densité du réseau dorsal virtuel

La Figure 4.16 montre l'effet de la densité du réseau dorsal virtuel sur le taux moyen de succès. Ce dernier facteur est encore fois étudié dans le but d'analyser l'équation (3.8) mais cette fois d'après l'indice du taux moyen de succès. La constatation est la même que celle déjà formulée. Il ne faut point que le nombre de nœuds appartenant dépasse une borne supérieure dans notre proposition, estimée à 30 nœuds.

La performance dégrade aussitôt le dépassement de cette limite. Cette affirmation certifie l'équation théorique. La dégradation est due principalement à la Phase II de la proposition, soit la maintenance. En effet, dans cette phase les nœuds s'envoient régulièrement des messages HEY pour s'assurer d'être en contact. Intuitivement, on réalise que plus la densité des nœuds stables est élevé, plus de messages doivent être échangés et traités, créant une latence supplémentaire au niveau de chaque nœud.

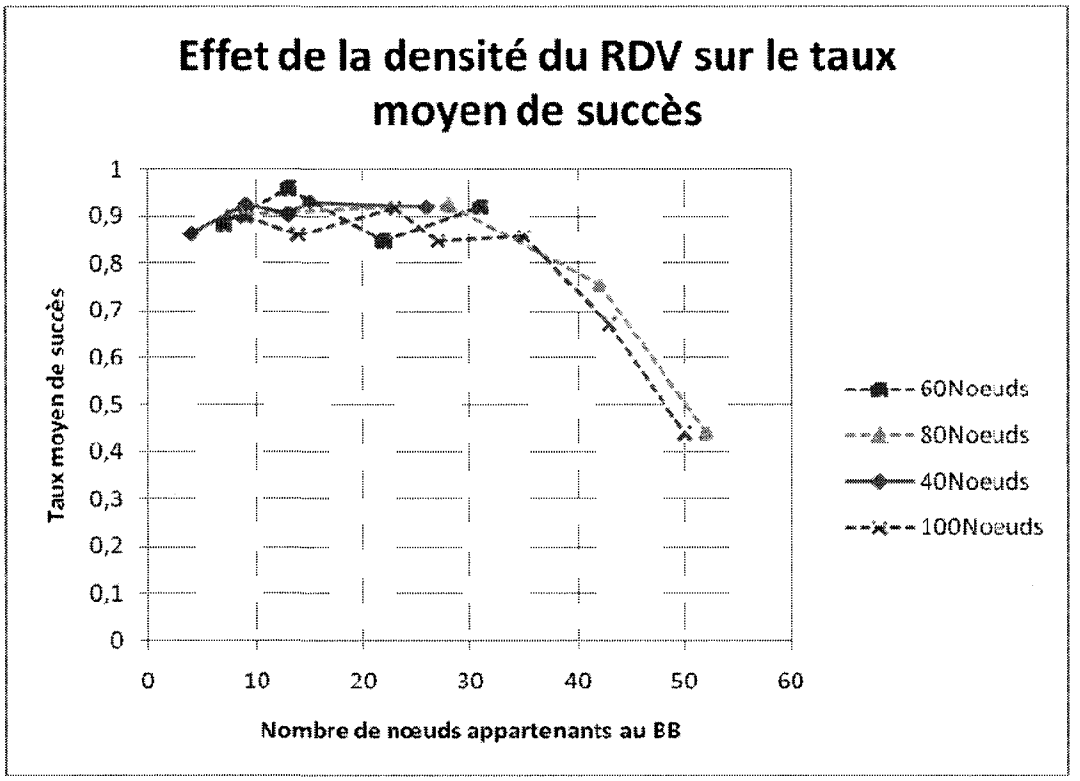


Figure 4.16 Effet du nombre de nœuds appartenant au RDV sur le taux moyen de succès

4.3.3 Influence des facteurs sur la charge du réseau

Effet de la mobilité

La Figure 4.17 (a) qui suit illustre l'effet de la mobilité sur le dernier indice de performance à l'étude, soit la charge du réseau. On aperçoit une légère dégradation de

performance pour une mobilité élevée car les déconnexions deviennent plus fréquentes. Dans le cas de pertes de liens, la phase II, soit la maintenance, est initiée seulement après que le protocole ait atteint un état indésirable, i.e. le nombre de nœuds appartenant au RDV est égal à la limite inférieure tolérée. Même si cette probabilité de maintenance est faible, à cause du critère de stabilité des nœuds du RDV, la reconstruction locale en tant que tel est légère. D'un autre côté, étant donné que seul le nœud du RDV le plus proche de l'UM répond par unicast aux enregistrements ou aux requêtes de services, le réseau est peu chargé de messages.

Nous savions intuitivement que l'apport majeur de notre proposition par rapport aux approches proposées dans la littérature est la diminution du nombre de paquets de signalisation ou de contrôle dans le réseau. Cette intuition a été confirmée par tous les facteurs analysés. La flagrance de la différence entre la charge de réseau de notre proposition avec celle de DSDP est montrée dans la Figure 4.17 (b).

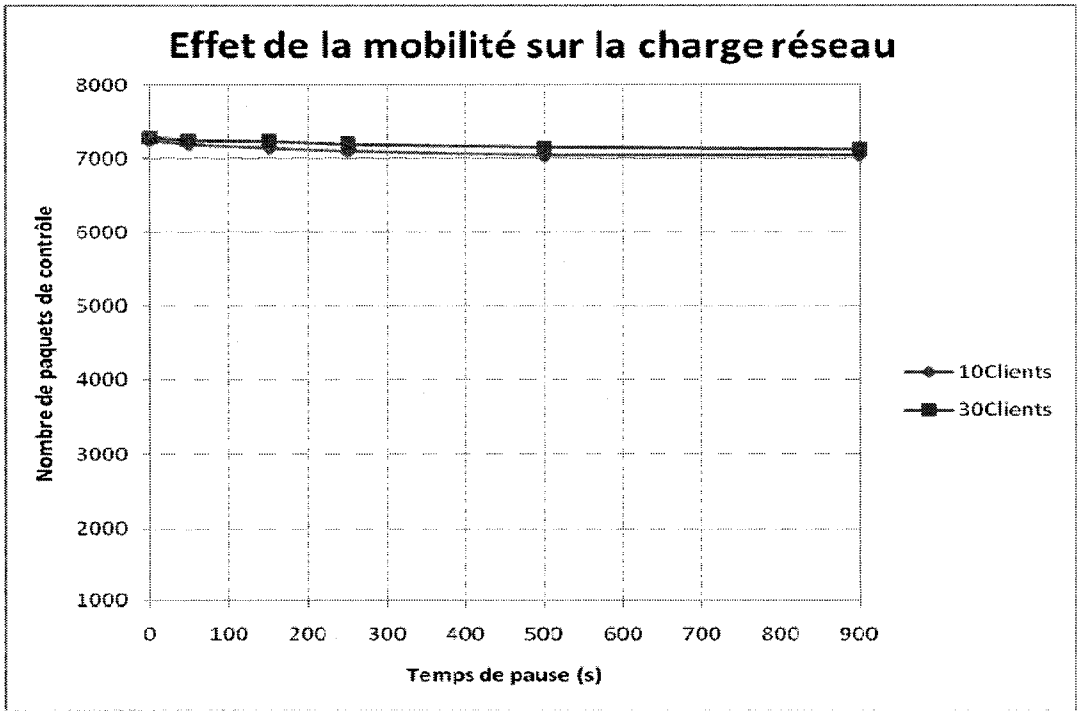


Figure 4.17 (a) Effet de la mobilité sur le nombre de paquets de contrôle

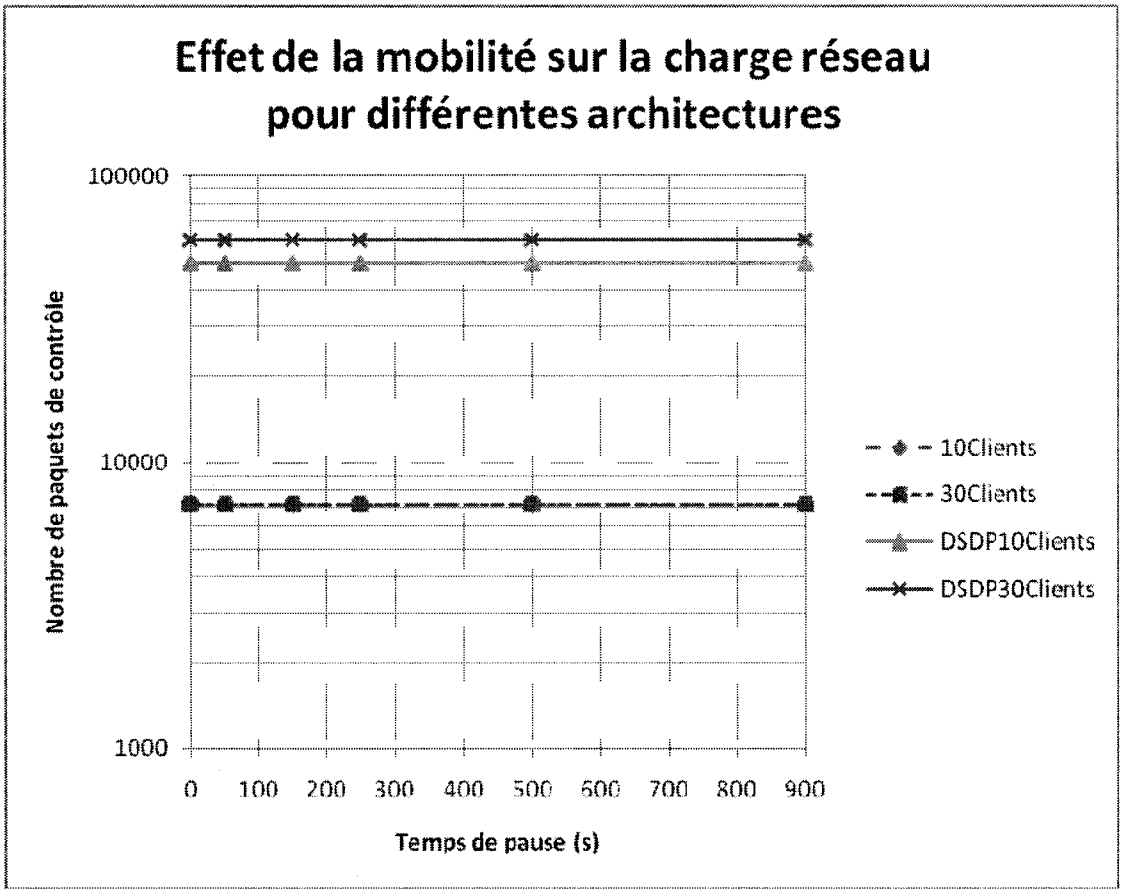


Figure 4.17 (b) Effet de la mobilité sur la charge pour différentes architectures

La contribution de notre proposition témoigne d'une économie de plusieurs dizaines de milliers de paquets de signalisation dans le réseau. Ceci est dû au fait que notre RDV est constitué de nœuds judicieusement choisis à l'intérieur de bornes ayant fait l'objet d'une analyse préalable. On conclut que l'approche est légère dans le sens où le réseau est peu chargé et constitué d'un ensemble minime de nœuds stables.

Effet du nombre de serveurs et de clients

Les Figures 4.18 et 4.19 qui suivent illustrent l'effet du nombre de serveurs et de clients sur la charge du réseau. Encore une fois, notre proposition montre de très bons résultats comparativement à l'approche DSDP.

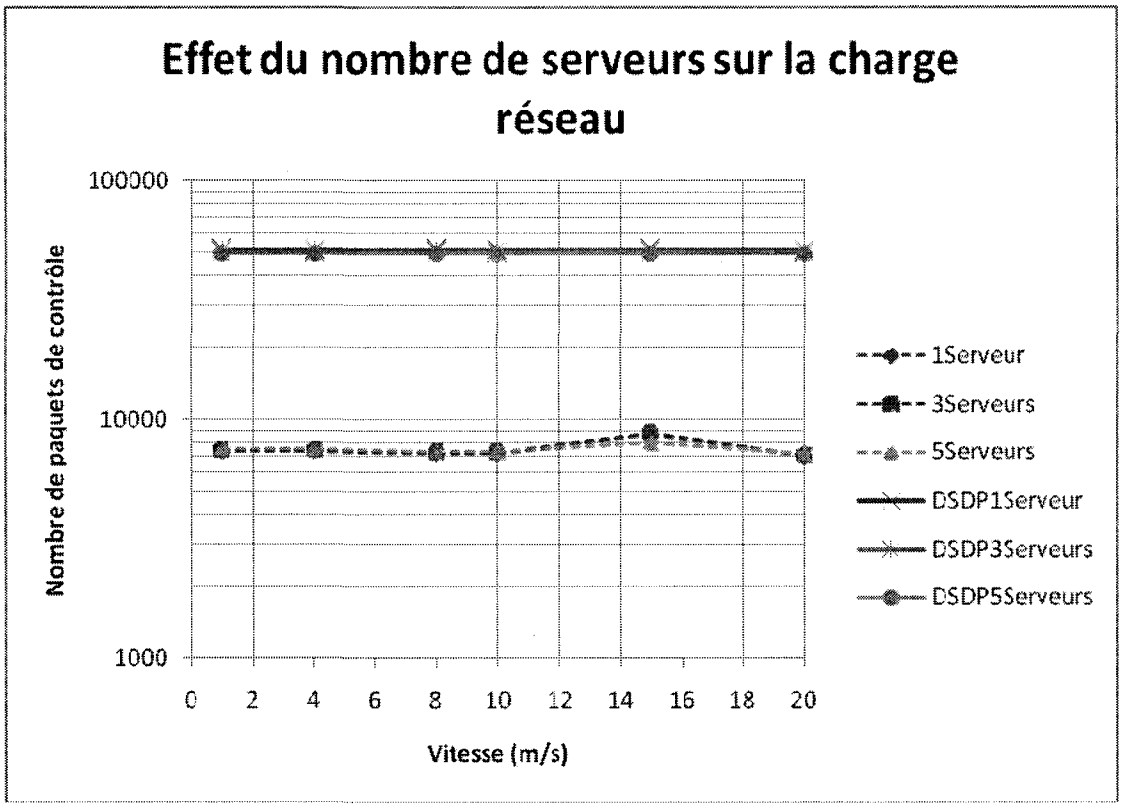


Figure 4.18 Effet du nombre de serveurs sur la charge réseau

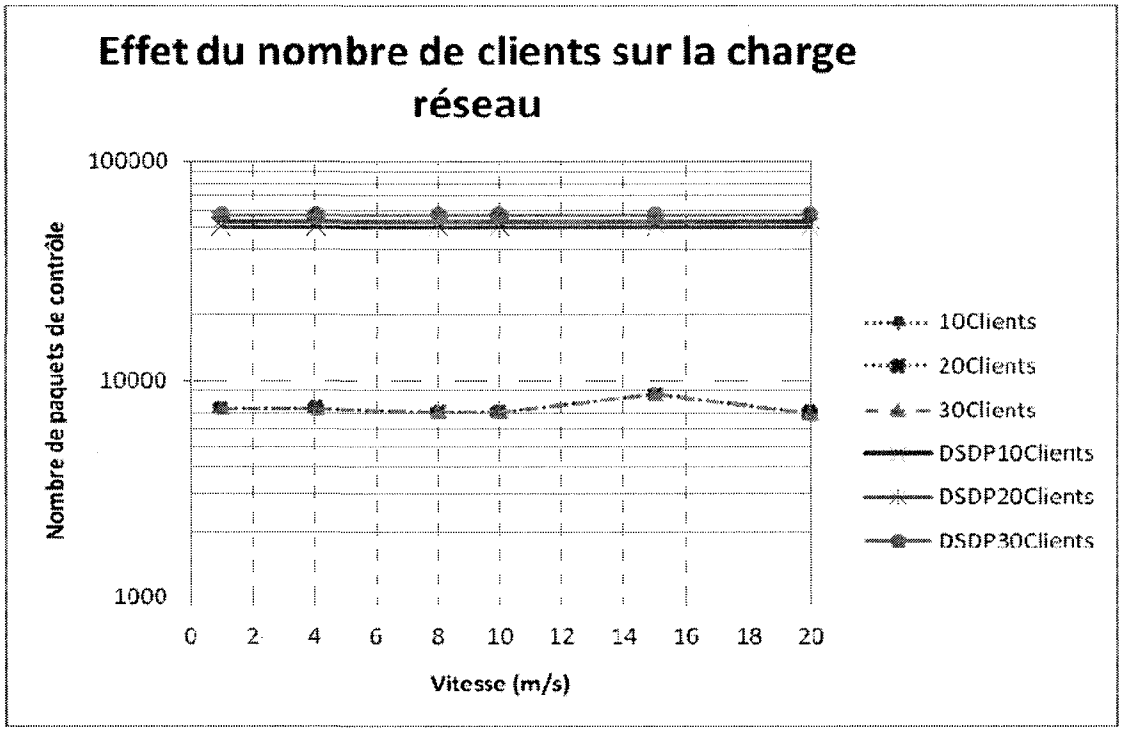


Figure 4.19 Effet du nombre de clients sur la charge réseau

4.5 Synthèse des performances

À la suite de la présentation et de l'analyse détaillée des résultats, une synthèse s'impose pour situer la solution proposée. Les résultats obtenus rencontrent des objectifs d'efficacité globalement très pertinents, notamment sur la charge du réseau en termes de nombre de paquets de contrôle. Le pourcentage de diminution du nombre de messages de signalisation est plus de 80% comparativement aux meilleures propositions dans la littérature. Ceci fait de notre proposition de découverte de service une solution légère mais améliorable. En effet, nous estimons avoir obtenus des délais de réponse aux requêtes raisonnables mais perfectible. À ce niveau, il faudrait pouvoir envisager une analyse particulière des paramètres utilisés pour le calcul du degré de stabilité au niveau de chaque nœud. Quant au taux moyen de succès aux requêtes, il a subit une amélioration de près de 1% en moyenne sur l'ensemble des cas considérés. Amélioration

minime mais considérant que les cas considérés avaient déjà de très bons taux de succès de l'ordre de 90% de tentatives réussites, nous pouvons marquer ce perfectionnement.

Dans tous les cas, le modèle proposé est intrinsèquement supérieures aux modèles existants, notamment quand un grand nombre de nœuds dans le réseau est envisagé, quand la mobilité est élevée, et même pour de grandes topologies de terrain. En somme, nous pouvons affirmer que l'évaluation de performances met en lumière la validité de notre approche adoptée pour assurer la découverte de services par le déploiement d'un nombre contingenté de nœuds stables dans le réseau, à moindre coût.

CHAPITRE V

CONCLUSION

Dans ce mémoire, nous avons traité du problème de la découverte de services dans les réseaux ad hoc. Dans de tels environnements décentralisés, les approches basées sur la définition d'un critère de stabilité d'un nœud et la distinction, parmi les unités mobiles du réseau, des nœuds considérés stables sont parues prometteuses. Ainsi, nous avons proposé une stratégie basée sur le déploiement d'un réseau dorsal virtuel (RDV). Il faut mentionner que l'objectif principal des nœuds du RDV est de maintenir la liste des services offerts dans le réseau dans le but de réduire la latence d'accès à l'information désirée, de diminuer la charge dans le réseau et d'améliorer le taux de succès de réponse à une requête. Dans ce chapitre, nous allons faire une synthèse des travaux réalisés. Nous allons également présenter les limites de la stratégie que nous avons proposée ainsi que quelques pistes pour des travaux futurs dans le domaine.

5.1 Synthèse des travaux

Dans ce projet, nous avons modifié la caractéristique de stabilité d'un nœud dans un MANET. En effet, la combinaison linéaire des paramètres tels que l'énergie de la batterie, la vitesse moyenne d'un nœud, l'espace de stockage, le nombre total de voisins directs ainsi que la plus longue période de résidence dans la portée de transmission de l'émetteur donne une bonne estimation de la stabilité d'un nœud. C'est sur la base de ces paramètres qu'on a considéré qu'une unité mobile est éligible d'être qualifiée de stable. Avant d'aller plus loin, il faut mentionner le constat suivant : le seul travail dans la littérature sur la sélection de nœuds stables dans le réseau pour la découverte de services s'est avéré peu réaliste. En effet, les auteurs de [19] supposent qu'un nœud sur deux est stable. Cette hypothèse est contestable car il importe de donner de la valeur au terme stable dans un environnement tel que les réseaux ad hoc. Ce terme ne peut être

associé qu'aux nœuds ayant un très haut niveau de fiabilité. C'est pourquoi nous avons esquissé une formulation mathématique de ce problème en mettant un accent particulier sur les contraintes auxquelles sont soumises les unités mobiles ainsi que le réseau ad hoc qu'elles constituent.

De façon spécifique, pour atteindre des objectifs de souplesse et prendre en compte des systèmes à grande échelle, notre architecture basée sur le déploiement d'un réseau dorsal virtuel pour repérer et enregistrer les services à l'intérieur d'un réseau dont la topologie change dynamiquement est apparue très prometteuse. Dans notre modèle, il s'agit de former un RDV même dans un état du réseau où la mobilité est très élevée en désignant des unités mobiles clés pour héberger la liste des services offerts dans le réseau. Ces unités disposant d'un profil bien déterminé sont les seuls à entrer dans le processus de découverte de services, les autres se contenteront uniquement, si c'est le cas, d'envoyer des offres ou des requêtes de services.

Notre proposition se décortique en trois phases. Dans la phase I, pendant un certain temps, tous les nœuds du réseau envoient et collectent des messages HELLO pour connaître leur voisinage afin d'associer une valeur de stabilité à chaque nœud dans l'environnement local. L'idée est de dégager au fur et à mesure de l'évolution du système les entités stables du réseau. La mobilité anarchique des entités du réseau ne peut permettre aux nœuds stables de rester en contact pendant un nombre indéfini de temps. Outre la mobilité, la consommation d'énergie et la durée de vie des liaisons sont d'autres risques qui mettent en péril les communications entre nœuds du RDV. Fort de ces restrictions liées aux comportements intrinsèques des réseaux ad hoc, nous avons envisagé une deuxième phase qui est la maintenance du RDV. En effet, la maintenance est locale de manière à réagir rapidement aux changements dans la topologie. Ainsi, si pour une raison ou pour une autre un nœud quitte le réseau dorsal soit par une déconnexion causée par un manque d'énergie ou à cause de son dynamisme, une reconstruction du chemin doit se faire localement. Ainsi le processus de reconstruction est mis en place dans la Phase II pour régler la situation localement en trouvant un autre nœud stable pour remplacer le nœud perdu. Dans la Phase III, nous avons présenté les

mécanismes pour permettre aux serveurs d'enregistrer leurs services et aux clients de prendre connaissance de l'emplacement des services qu'ils désirent.

Notre modèle a fait l'objet d'une comparaison avec le modèle DSDP dans lequel toutes les unités mobiles participent au processus de découverte de service. Pour évaluer les performances du modèle que nous avons proposé, nous avons implémenté et simulé la stratégie au moyen d'un simulateur pour réseau sans fil.

Des indices de performance ont été définis ainsi que des facteurs qui influencent le modèle. Les variations de ces facteurs à travers des niveaux préfixés ont fourni un ensemble de résultats montrant la performance du modèle de découverte de service, comparativement aux autres configurations étudiées. Dans la majeure partie des cas étudiés, la charge de réseau est nettement réduite, le taux de succès de réponse aux requêtes est légèrement amélioré et le délai moyen de réponse aux requêtes est satisfaisant comparativement aux meilleurs délais notés dans la littérature. Le déploiement d'un réseau dorsal virtuel pour la découverte de services, jusque-là quasi inexplorée, est donc une stratégie dont l'usage est à approfondir dans ces types de réseaux.

5.2 Limites de l'approche proposée

En dépit des résultats intéressants obtenus avec la stratégie de déploiement d'un réseau dorsal virtuel pour la découverte de services proposée dans ce mémoire, il persiste certaines limitations sur lesquelles nous mettons l'emphase. En effet, les résultats obtenus qui assurent la performance de notre stratégie par rapport aux autres approches sont générés avec l'usage du protocole de routage DSR. Les mêmes expériences effectuées avec un autre protocole tel qu'AODV produisent une dégradation de performance de la stratégie proposée en ce qui concerne le délai moyen de réponse aux requêtes. Donc, le problème de routage constitue toujours un goulot d'étranglement dans les réseaux ad hoc. Le modèle de découverte de service proposé n'est donc pas indépendant du protocole de routage utilisé. Il faudra donc, pour le rendre indépendant

du protocole de routage, effectuer des améliorations en tenant compte des réalités du routage dans les réseaux ad hoc.

5.3 Travaux futurs

La stratégie proposée pour le déploiement d'un réseau dorsal virtuel basé sur le profil des nœuds du réseau a donné des résultats suffisamment prometteurs pour donner lieu à des développements ultérieurs. Les résultats d'expérimentation ont révélé une piste de considération assez importante. Le délai moyen de réponse à une requête est principalement causé par le processus de maintenance introduit à la phase II de la proposition. En effet, il s'agirait de gérer convenablement les échanges de messages entre nœuds du RDV. Plus généralement, notre modèle pourrait être, sous certaine condition, combiné à une procédure de graphe de recouvrement pour minimiser la latence de transfert de messages.

Notre modèle suppose que l'intégration du protocole de découverte de service au protocole de routage sous-jacent n'aura intégration pas un impact important sur la charge du réseau. Il pourrait être très intéressant d'étudier et de concevoir un mécanisme qui fusionne les protocoles en question afin de vérifier si la charge du réseau ne pourrait être diminuée davantage.

Enfin, il serait aussi pertinent d'ajouter d'autres paramètres à la combinaison linéaire qui définit la stabilité d'un nœud. Ce modèle analytique pourrait être davantage augmenté de critères permettant de contingenter le nombre de nœuds stables du réseau et par ce fait, représenter encore mieux les caractéristiques intrinsèques des unités mobiles d'un réseau ad hoc.

BIBLIOGRAPHIE

[1] D. Tang, C. Chang, K. Tanaka, M. Baker, "Resource discovery in ad hoc networks," Tech. Rep., Stanford University, August 1998, CSI-TR-98-769

[2] D. S. J. De Couto, D. Aguayo, B. A. Chambers, and R. Morris, "Performance of multihop wireless networks: shortest path is not enough," *ACM SIGCOMM Computer Communication Review*, Vol. 33, No. 1, pp. 83_88, january 2003

[3] D. Chakraborty, A. Joshi, and Y. Yesha, "Integrating service discovery with routing and service management for ad-hoc networks," *Ad Hoc networks Journal*, Vol. 4, No 2, pp. 204-224, Mars 2006

[4] DARPA Agent Markup Language, http://www.daml.org

[5] D. Chakraborty, A. Joshi, "GSD: A Novel Group-based Service Discovery Protocol for MANETS," *Proceedings of the 4th IEEE Conference on Mobile and Wireless Communications Networks*, pp. 140-144, July 2002

[6] Y. Zhang and L. Cheng, PLACE, "Protocol for location and coordinates estimation -- a wireless sensor network approach," *Computer Networks* (Elsevier), Vol. 46, No. 5, pp. 670-693, December 2004

[7] E. W. Zegura, M. H. Ammar, et al. "Application-layer Anycasting: A server Selection Architecture and Use in a replicatedWeb Service," *IEEE/ACM Transactions Networking*, Vol. 30, No. 8, pp. 455-466, 2000

[8] D.B. Johnson and D. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Network," *Mobile Computing,* Chapitre 5, pp. 88-153, 1996

[9] RFC 3561 - Ad hoc On-Demand Distance Vector (AODV) Routing, Juillet 2003, http://www.faqs.org/rfcs/rfc3561.html

[10] E.M Royer, C-K Toh, "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks," *IEEE Personal Communications*, pp. 46-55, April 1999

[11] Sun Microsystems, JINI Architecture Specification, Nov. 1999

[12] E. Guttman, C. Perkins, J. Veizades, and M. Day, "Service Location Protocol, Version 2," *IETF RFC 2608*, June 1999

[13] M. Barbeau, "Service Discovery Protocols for Ad Hoc Networking," *CASCON 2000 Workshop on ad hoc communications*, 2000

[14] S. Chien-Chung, S. Thapornphat, R. Liu, Z. Huang, "CLTC: a cluster-based topology control for ad hoc networks," *Transactions on Mobile Computing*, Vol. 3, No. 1, pp. 18-32, January 2004

[15] D. Meyer, "Administratively scoped IP Multicast," IETF RFC 2365, July 1998

[16] K. Chakraborty, S. S. Iyengar, H. Qi, E. Cho, "Grid Coverage for Surveillance and Target Location in Distributed Sensor Networks," *IEEE Transactions Computers*, Vol. 51, No. 12, pp. 1448 -1453, 2002

[17] H. Koubaa, L. Inria Lorraine, "Un protocole de couverture de services dans les réseaux ad hoc sans-fil,"
http://www.ares.insa-lyon.fr/tarot/download/**HendKoubaa**_03_01.ppt

[18] R. Sivakumar, P. Sinha, V. Bharghavan, "A core-extraction distributed ad hoc routing algorithm," *IEEE Journal on selected Areas in Communications*, Vol. 17, No. 8, pp. 1454-1465, 1999

[19] Ulas C. Kozat, G. Kondylis, B. Ryu, K. Marwma, "Virtual Dynamic Backbone for mobile ad hoc networks," *Proceedings of IEEE International Conference on Communications*, Vol.1, pp. 250-255, June 2001

[20] Christopher Dabrowski, Kevin Mills, "Understanding Self-healing in Service-Discovery Systems," *In ACM Workshop on Self-Healing systems*, Charleston SC USA, 2002, http://www.itl.nist.gov/div897/cgt/adl/sdp_projectpage.html

[21] Qualnet Simulator
http://scalable-networks.com/help/index.html

[22] R. Sivakumar, P. Sinha, and V. Bharghavan, "CEDAR: A core-extraction distributed ad hoc routing algorithm," *Journal on Selected Areas in Communications*, Vol. 17, No. 8, pp. 1454–1465, August 1999

[23] H. Luo, M. Barbeau, "Performance Evaluation of Service Discovery Strategies in Ad Hoc Networks," *Second Annual Conference on Communication Networks and Services Research* (CNSR'04), pp. 61-68, 2004